UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Ingeniería de Telecomunicación: Telemática

PROYECTO FIN DE CARRERA

DESARROLLO DE UNA HERRAMIENTA WEB 2.0 DE ANOTACIÓN Y BÚSQUEDA DE PARTIDAS DE AJEDREZ EN BASES DE DATOS ESPECIALIZADAS

Autor: Juan José del Peral Pérez
Tutor: Jesús Arias Fisteus

NOVIEMBRE DE 2009

Agradecimientos

Gracias a Marta, por su inapreciable ayuda, ánimo, amor y paciencia, sin los que no hubiera podido concluir este proyecto.

Gracias a mis padres, por su apoyo y su amor incondicionales y por los enormes sacrificios realizados para mi educación.

Gracias también a mi hermano por su cariño y su amistad y a Fernando por su valiosa ayuda.

Gracias a David, mi gran compañero y amigo, por tantos momentos y por enseñarme a trabajar en equipo en particular y a trabajar en general. También a Irene, Laura, Rocío y Sara.

Y por último gracias a mis profesores. A Emilio, por "obligarme" a estudiar Analógica. A Francisco, por su tiempo y dedicación durante años; siento no haber llegado hasta aquí con él. Y a Jesús, por toda su ayuda a pesar de la enorme distancia.

Resumen del proyecto

En la Web se encuentran disponibles diversas bases de datos de partidas de ajedrez de acceso libre, pero la capacidad de búsqueda de información en ellas se reduce a metadatos como nombre de los jugadores, fecha, apertura, etc. Portable Game Notation (PGN) es el estándar de facto para el almacenamiento de partidas en formato texto, por lo que la mayoría de estas bases de datos proporcionan las partidas en dicho formato. Por otra parte, también existen aplicaciones de visualización de partidas de código libre capaces de reproducir partidas almacenadas en formato PGN. El objetivo de este proyecto es desarrollar un sistema de Web social que integre un visor de partidas de código libre ya existente y el acceso a una o más bases de datos de partidas. La innovación de este sistema está en permitir a los usuarios realizar anotaciones mediante etiquetas (tags) de partidas y fragmentos de ellas (una posición, una jugada, una secuencia de jugadas). Esto permitiría a los usuarios del sistema encontrar partidas de forma individualizada, a partir de las etiquetas asociadas a las partidas, además de mediante los campos tradicionales. A su vez implementaría una de las características de la Web 2.0, como es el etiquetado.

Tabla de contenidos

Capítulo 1

Introducción

1.1 Motivación

El mundo actual está en constante y frenética evolución. La dificultad de mantenerse al día con cada nueva tecnología es cada día mayor, ya que la velocidad con que van quedando obsoletas es simplemente pasmosa. Un buen ejemplo es la misma Internet. Pese a su relativa juventud, Internet ha evolucionado y sigue haciéndolo de manera sorprendente. Forma parte prácticamente imprescindible de la vida de la mayoría de los seres humanos, de modo que resulta difícil imaginar la vida moderna si no existiese Internet. Últimamente está en boca de todos el concepto de web 2.0. Enriquecer la red por medio de aquéllos que hacen uso de ella. Ceder control al usuario, de modo que se haga colectivo el beneficio del conocimiento, pensamiento y esfuerzo individual. Es muy interesante la potencia que puede llegar a tener Internet, si es utilizada de manera óptima, para el aprendizaje y el bien comunes. Como es interesante, a su vez, el contraste de esta evolución con un deporte como es el ajedrez.

El ajedrez es un juego que ha permanecido prácticamente inmutable durante los últimos quinientos años, que ha apasionado a personajes como Voltaire, Rousseau, Benjamin Franklin, Napoleón Bonaparte, Robespierre, Catalina la Grande o Goethe. Aún hoy continúa siendo uno de los juegos más difundidos por todo el mundo, también por medio de las nuevas tecnologías: ajedrez electrónico, partidas en línea, aplicaciones de escritorio y aplicaciones web para reproducir partidas. Es este último uso en el que se basa este proyecto, en la posibilidad de reproducir una partida de ajedrez almacenada en un formato determinado, mediante una herramienta web.

Hay muchas aplicaciones que así lo hacen, pero ninguna que utilice la red según el concepto web 2.0. Por tanto, resulta muy atrayente la idea de que los propios usuarios puedan compartir su conocimiento con otros aficionados al ajedrez mediante una herramienta que permita dicha interacción. A tal efecto, uno de los mecanismos de la web 2.0, el etiquetado, parece especialmente apropiado como complemento para los visores de partidas de ajedrez.

Una etiqueta puede definirse como una palabra o frase generada por un usuario con el fin de organizar el contenido web y definirlo de una forma más humana, lo que crea una mayor identificación con dicho contenido. O como lo describió David Weinberger, un filósofo cuyo trabajo se orienta hacia la relación entre el ser humano e Internet, en una entrevista para the Pew Internet & American Life Project [1]: “Quizás lo más interesante sobre el etiquetado es que ahora tenemos a millones y millones de personas que están diciendo, en público, lo que piensan sobre páginas e imágenes”.

En una partida de ajedrez, estas etiquetas permitirían que los usuarios pudieran expresar cualquier comentario y pensamiento sobre fragmentos de una partida. Ejemplos de lo que podría ser etiquetado son: un error cometido por uno de los jugadores que fuera determinante en la partida, los nombres de las aperturas utilizadas, movimientos especialmente interesantes, particularidades o información específica sobre la partida, etc.

En resumen, resulta tremendamente interesante utilizar el concepto de web 2.0 para potenciar el intercambio de información mediante partidas de ajedrez, y por tanto la implementación de una herramienta que lo haga posible, mediante el uso de etiquetas.

1.2 Objetivos

El objetivo de este proyecto será, por tanto, la implementación de una aplicación de visualización de partidas de ajedrez que permita realizar etiquetado de los contenidos de la misma, rigiéndose así por el concepto de web 2.0. La implementación se desarrollará en el lenguaje de programación Java, uno de los lenguajes de programación más extendidos. El hecho de ser un lenguaje multiplataforma y poseer librerías muy potentes para implementar interfaces gráficas, hace que sea sumamente adecuado para el desarrollo de la aplicación.

Por otro lado, la información debe almacenarse en una base de datos, por lo que puede que sea necesario el diseño y la implementación de una base de datos que cumpla con nuestros requerimientos si no se encuentra una base de datos específica ya existente para tal efecto.

También será necesario diseñar un modo de poblar dicha base de datos, para lo cual se desarrollará una herramienta que permita introducir partidas en la base de datos elegida.
Ese es el objetivo principal del proyecto. Pero el hecho de ser introducidas las etiquetas por usuarios plantea el problema de su gestión: la aplicación debe permitir gestionar de alguna forma dichos usuarios, aunque no sea el objetivo principal de este proyecto. Por lo tanto se optará por implementar una gestión sencilla de los mismos, permitiendo a un usuario abrir una cuenta en la aplicación e introducir etiquetas en las partidas almacenadas en la base de datos. También se debe permitir visualizar de alguna forma la información introducida por el resto de usuarios.

En resumen, la aplicación objeto de este proyecto deberá permitir el acceso de un usuario a una herramienta que permita visualizar partidas de ajedrez almacenadas en una base de datos, añadir etiquetas a las mismas, y acceder a las etiquetas de otros usuarios.

1.3 Plan de trabajo

Antes de comenzar con la implementación de la herramienta, lo más conveniente es un estudio previo de las herramientas ya existentes, debido a la posibilidad de utilizar las

funcionalidades de alguna de ellas y ampliarlas con las funcionalidades requeridas por el proyecto.

El primer paso, por tanto, consistirá en un estudio exhaustivo de las herramientas para la visualización de partidas y de las bases de datos de partidas de ajedrez ya existentes. También deben considerarse las opciones de tecnología para la implementación de la aplicación, así como los distintos formatos de partidas y tipos de licencias que conciernan a las posibles herramientas a utilizar.

Posteriormente, se presentarán a priori dos alternativas fundamentales: utilizar una herramienta ya existente y adaptarla, o bien realizar una implementación a medida. Esta decisión deberá ser tomada, fundamentalmente, tanto para la aplicación de visualización de partidas como para la base de datos. Si se necesitara una herramienta para poblar la base de datos, muy probablemente deberá ser implementada.

El paso siguiente debe ser la elección de un patrón de diseño, tanto para la base de datos como para la herramienta de visualización. Si se ha decidido reutilizar alguna aplicación, el diseño estará marcado por la herramienta preexistente. De lo contrario, se deberá realizar un diseño integral que cumpla con los requisitos del proyecto. Debe tenerse en cuenta que el diseño y elección de la base de datos tendrá un impacto fundamental en la aplicación, de modo que se elegirá un diseño adecuado, que facilite en la medida de lo posible el desarrollo de la herramienta.

El paso lógico siguiente, ya que las partidas de la base de datos son un recurso fundamental de la aplicación de visualización, será comenzar a desarrollar la base de datos, bien implementando o ampliando una existente.

Una vez implementada la base de datos, se debe proceder a introducir información referente a partidas en ella. Si no se hallase una herramienta satisfactoria para tales efectos, se procederá a su implementación. El diseño se buscará lo más sencillo posible, ya que la única función de la herramienta es poblar la base de datos y no estará integrada en la aplicación principal del proyecto. Se realizarán pruebas en la herramienta para comprobar su correcto funcionamiento y la integridad de la información introducida en la base de datos.

Una vez se cuente con una base de datos poblada de partidas, se podrá comenzar con la implementación de la herramienta o de los cambios en la herramienta elegida, si se optara por utilizar una ya existente. Se implementarán mecanismos que permitan a la herramienta conectar con la base de datos, extraer partidas y desplegarlas. A su vez se implementará la funcionalidad clave del proyecto, la introducción de etiquetas asociadas a fragmentos de partidas, y la obtención de la información introducida por el resto de usuarios mediante etiquetas. Se procederá también a la realización de pruebas que aseguren el correcto funcionamiento de la aplicación, así como la integridad de la información introducida por los usuarios.

Por último se deberá implementar una gestión básica de los usuarios, permitiendo operaciones tales como abrir una cuenta y acceder a ella, lo que permitirá por consiguiente acceder a la aplicación. Se realizarán pruebas sobre este módulo también para comprobar su correcto funcionamiento.

1.4 Organización de la memoria

En este apartado se pasará a comentar brevemente la estructura del presente documento y el contenido de los subsiguientes apartados:

- Estado del Arte: en este apartado se analizarán las distintas opciones tecnológicas existentes actualmente y en las que el presente proyecto deberá basarse. Entre ellas se encuentran la oferta de sistemas de gestión de bases de datos, las bases de datos de partidas de ajedrez presentes en la red y las distintas herramientas de visualización de partidas de ajedrez, con sus correspondientes tecnologías. También se considerarán las particularidades del formato de representación de partidas de ajedrez PGN (Portable Game Notation), el más extendido con enorme diferencia, así como una descripción del concepto de web 2.0, fundamental en el proyecto y de las tecnologías implicadas en ello.

- Requisitos: este apartado enumerará los requisitos, tanto funcionales como no funcionales de la aplicación, agrupados por su similitud.

- Arquitectura de la aplicación: análisis de las distintas alternativas de diseño inferidas a partir de los requisitos de la aplicación. Se escogerá una de las posibilidades como la más adecuada, argumentando los motivos que lleven a tal conclusión.

- Diseño de la interfaz gráfica: explicación argumentada del diseño de la interfaz gráfica para el visualizado de las partidas. Si se reutilizara una aplicación ya existente, se deberían exponer las razones en este mismo apartado, así como las modificaciones a las que se diera lugar.

- Diseño: en este apartado se pasarán a explicar los detalles concernientes al diseño de la aplicación. Dicho diseño incluye tanto el diseño de base de datos, si fuere necesario, como el diseño de los distintos módulos que pudieran componer la aplicación. Se incluirán diagramas y gráficos si se hallase conveniente.

- Implementación: en este apartado se analizarán particularidades de la implementación que fuesen determinantes para la comprensión del funcionamiento del programa. Se incluirán fragmentos de código a modo de apoyo de las explicaciones.

- Pruebas: descripción del plan de pruebas al que habrá sido sometido el programa con el fin de comprobar su correcto funcionamiento de acuerdo a las especificaciones.

- Conclusiones y Trabajos Futuros: en este apartado se procederá a un análisis a posteriori del proyecto, evaluando la consecución de los objetivos planteados inicialmente, la idoneidad de las tecnologías seleccionadas y del diseño realizado. Se comentarán posibles problemas surgidos durante todo el proceso anteriormente descrito y, como trabajos futuros, se propondrán posibles mejoras o funcionalidades adicionales que pudieran derivarse del presente proyecto. También se podrán proponer proyectos futuros que pudieran basarse en el trabajo ya realizado.

- Apéndices: se incluirán a modo de apéndice, ciertos aspectos relativos al proyecto, como pudieran ser los siguientes:

    - Presupuesto: estimación tanto en tiempo como en gasto del proceso integral de generación del proyecto.
    - Manual de instalación y usuario: explicación básica tanto de los pasos de instalación como del funcionamiento de la aplicación desde el punto de vista de un usuario.
    - Contenido adicional asociado a alguno de los apartados de la documentación, como pudieran ser fragmentos de código o tablas explicativas.

Capítulo 2

Estado del Arte

En la elaboración de un proyecto confluyen diversas tecnologías y herramientas para su implementación. Hay un extenso abanico de opciones. Por tanto es conveniente analizar las más importantes y deducir sus ventajas e inconvenientes principales. También es necesario analizar ciertos concepto en que está basado el proyecto y que deben influir en el posterior diseño.

2.1 Web 2.0

El concepto original del contexto, llamado Web 1.0 era páginas estáticas HTML (HyperText Markup Language) que no eran actualizadas frecuentemente. El éxito de las punto-com dependía de webs más dinámicas (a veces llamadas Web 1.5) donde los Sistemas de gestión de contenidos servían páginas HTML dinámicas creadas al vuelo desde una actualizada base de datos. En ambos sentidos, el conseguir visitas y la estética visual eran considerados como factores importantes.

Según wikipedia [2] los propulsores de la aproximación a la Web 2.0 creen que el uso de la web está orientado a la interacción y redes sociales, que pueden servir contenido que explota los efectos de las redes, creando o no webs interactivas y visuales. Es decir, los sitios Web 2.0 actúan más como puntos de encuentro, o webs dependientes de usuarios, que como webs tradicionales.

Al comparar la Web 2.0 con la 1.0 vemos que la última principalmente trata lo que es el estado estático, es decir los datos que se encuentran en esta no pueden cambiar, se encuentran fijos, no varían, no se actualizan. De acuerdo con Tim O'Reilly [3], la Web 2.0 puede ser comparada con la Web 1.0 de esta manera:

Web 1.0	Web 2.0
DoubleClick	Google AdSense
Ofoto	Flickr
Akamai	BitTorrent
mp3.com	Napster
Britannica Online	Wikipedia
personal websites	blogging
evite	upcoming.org and EVDB
domain name speculation	search engine optimization
page views	cost per click

screen scraping	web services
publishing	participation
content management systems	wikis
directories (taxonomy)	tagging ("folksonomy")
stickiness	syndication

En ocasiones se ha relacionado el término Web 2.0 con el de Web semántica. Sin embargo, ambos conceptos corresponden más bien a estados evolutivos de la web, y la Web semántica correspondería en realidad a una evolución posterior, a la Web 3.0 o web inteligente. La combinación de sistemas de redes sociales, como FOAF y XFN, con el desarrollo de etiquetas (o tags), que en su uso social derivan en folcsonomías, así como el plasmado de todas estas tendencias a través de blogs y wikis, confieren a la Web 2.0 un aire semántico sin serlo realmente. Sin embargo, en el sentido más estricto para hablar de Web semántica, se requiere el uso de estándares de metadatos como Dublin Core y en su forma más elaborada de ontologías y no de folcsonomías. De momento, el uso de ontologías como mecanismo para estructurar la información en los programas de blogs es anecdótico y sólo se aprecia de manera incipiente en algunos wikis.

Por tanto podemos identificar la Web semántica como una forma de Web 3.0. Existe una diferencia fundamental entre ambas versiones de web (2.0 y semántica) y es el tipo de participante y las herramientas que se utilizan. La 2.0 tiene como principal protagonista al usuario humano que escribe artículos en su blog o colabora en un wiki. El requisito es que además de publicar en HTML emita parte de sus aportaciones en diversos formatos para compartir esta información como son los RSS, Atom, etc. mediante la utilización de lenguajes estándares como el XML (eXtensible Markup Language). La Web semántica, sin embargo, está orientada hacia el protagonismo de procesadores de información que entiendan de lógica descriptiva en diversos lenguajes más elaborados de metadatos como SPARQL, POWDER u OWL que permiten describir los contenidos y la información presente en la web, concebida para que las máquinas "entiendan" a las personas y procesen de una forma eficiente la avalancha de información publicada en la Web.

El término Web 2.0 fue acuñado por Dale Dougherty de O'Reilly Media en una lluvia de ideas con Craig Cline de MediaLive para desarrollar ideas para una conferencia. Dougherty sugirió que la web estaba en un renacimiento, con reglas que cambiaban y modelos de negocio que evolucionaban. Dougherty puso ejemplos en vez de definiciones, y reclutó a John Battelle para dar una perspectiva empresarial, y O'Reilly Media, Battelle, y MediaLive lanzó su primera conferencia sobre la Web 2.0 en Octubre del 2004. La segunda conferencia se celebró en octubre de 2005.
En 2005, Tim O'Reilly definió el concepto de Web 2.0.

En su conferencia, O'Reilly y Battelle resumieron los principios clave que creen que caracterizan a las aplicaciones web 2.0: la web como plataforma; datos como el "Intel Inside"; efectos de red conducidos por una "arquitectura de participación"; innovación y

desarrolladores independientes; pequeños modelos de negocio capaces de redifundir servicios y contenidos; el perpetuo beta; software por encima de un solo aparato.

En general, cuando mencionamos el término Web 2.0 nos referimos a una serie de aplicaciones y páginas de Internet que utilizan la inteligencia colectiva para proporcionar servicios interactivos en red dando al usuario el control de sus datos.

Así, Xavier Ribes define como 2.0 -"todas aquellas utilidades y servicios de Internet que se sustentan en una base de datos, la cual puede ser modificada por los usuarios del servicio, ya sea en su contenido (añadiendo, cambiando o borrando información o asociando datos a la información existente), pues bien en la forma de presentarlos, o en contenido y forma simultáneamente." [4]

La infraestructura de la Web 2.0 es muy compleja y va evolucionando, pero incluye el software de servidor, redifusión de contenidos, protocolos de mensajes, navegadores basados en estándares, y varias aplicaciones para clientes.

Una web se puede decir que está construida usando tecnología de la Web 2.0 si se caracteriza por las siguientes técnicas:

- CSS, marcado XHTML válido semánticamente y Microformatos
- Técnicas de aplicaciones ricas no intrusivas (como AJAX)
- Java Web Start
- XUL
- Redifusión/Agregación de datos en RSS/ATOM
- URLs (Uniform Resource Locator) sencillas con significado semántico
- Soporte para postear en un blog
- JCC y APIs REST o XML
- JSON
- Algunos aspectos de redes sociales
- Mashup (aplicación web híbrida)

O también cumplir las siguientes reglas generales:

- El sitio no debe actuar como un "jardín cerrado": la información debe poderse introducir y extraer fácilmente
- Los usuarios deberían controlar su propia información
- Basada exclusivamente en la Web: los sitios Web 2.0 con más éxito pueden ser utilizados enteramente desde un navegador

Los protocolos de mensajes bidireccionales son uno de los elementos clave de la infraestuctura de la Web 2.0. Los dos tipos más importantes son los métodos RESTful y SOAP. REST indican un tipo de llamada a un servicio web donde el cliente transfiere el estado de todas las transacciones. SOAP y otros métodos similares dependen del servidor para retener la información de estado. En ambos casos, el servicio es llamado desde un API (Application Programming Interface). A veces este API está personalizado en

función de las necesidades específicas del sitio web, pero los APIs de los servicios web estándares (como por ejemplo escribir en un blog) están también muy extendidos. Generalmente el lenguaje común de estos servicios web es el XML, si bien puede haber excepciones.

Recientemente, una forma híbrida conocida como Ajax ha evolucionado para mejorar la experiencia del usuario en las aplicaciones web basadas en el navegador. Esto puede ser usado en webs propietarias (como en Google Maps) o en formas abiertas utilizando un API de servicios web.

La funcionalidad de Web 2.0 se basa en la arquitectura existente de servidor web pero con un énfasis mayor en el software dorsal. La redifusión sólo se diferencia nominalmente de los métodos de publicación de la gestión dinámica de contenido, pero los servicios Web requieren normalmente un soporte de bases de datos y flujo de trabajo mucho más robusto y llegan a parecerse mucho a la funcionalidad de internet tradicional de un servidor de aplicaciones. El enfoque empleado hasta ahora por los fabricantes suele ser bien un enfoque de servidor universal, el cual agrupa la mayor parte de la funcionalidad necesaria en una única plataforma de servidor, o bien un enfoque plugin de servidor Web con herramientas de publicación tradicionales mejoradas con interfaces API y otras herramientas. Independientemente del enfoque elegido, no se espera que el camino evolutivo hacia la Web 2.0 se vea alterado de forma importante por estas opciones.

2.2 Formatos de partidas

Es común encontrar distintas opciones de formato a la hora de representar un objeto en formato digital, como ocurre, por ejemplo, con el contenido multimedia. Con respecto a las partidas de ajedrez, existe un formato que es aceptado por unanimidad por todas las herramientas de visualización de partidas de ajedrez, el formato PGN. Existen otros formatos, que suelen ser dependientes de cierta herramienta o base de datos, por lo que no se considerará necesario su análisis.

Formato PGN

Wikipedia define el PGN [5] (Portable Game Notation) como un formato de computadora para grabar partidas de ajedrez, tanto los movimientos como la información relacionada; la mayoría de los programas de ajedrez para computadora reconocen este formato que es muy popular como consecuencia de su fácil uso.

El formato PGN está estructurado para una fácil lectura y escritura por usuarios humanos y para fácil análisis y generación por programas informáticos. Las jugadas están dadas en notación algebraica de ajedrez. Por lo general la extensión asignada a los archivos con este formato es ".pgn".

Existen dos subformatos dentro de la especificación del PGN, el formato de importación y el de exportación. El formato de importación describe información que ha sido preparada a mano y es intencionalmente flexible; un programa que pueda leer datos de un formato PGN debe ser capaz de manejar este formato flexible. El formato de exportación es en cambio estricto, describe la información generada bajo el control de un programa informático. El formato de exportación generado por distintos programas debe ser exactamente equivalente byte por byte.

El código informático del formato PGN empieza con un conjunto de pares de etiquetas (el nombre de la etiqueta y su valor), seguido de las jugadas (los movimientos del ajedrez con comentarios opcionales).

Cada par de etiquetas comienza con un "[", seguido del nombre de la etiqueta, el valor de la etiqueta encerrado en comillas dobles ("), y un "]" para cerrar.

Para almacenar la información en el formato PGN es necesario dar siete etiquetas, llamadas "STR" (del inglés Seven Tag Roster) que significa "lista de siete etiquetas". En el formato de exportación, las etiquetas STR deben aparecer antes que cualquier otro par de etiquetas. El orden es el siguiente:

1. Event: el nombre del torneo o de la competición.
2. Site: el lugar donde el evento se llevó a cabo. Esto debe ser en formato "Ciudad, Región PAÍS", donde PAÍS es el código del mismo en tres letras de acuerdo a l código del Comité Olímpico Internacional. Cómo ejemplo: "México, D.F. MEX".
3. Date: la fecha de inicio de la partida en formato AAAA.MM.DD. Cuando se desconocen los valores se utilizan "??".
4. Round: La ronda original de la partida.
5. White: El jugador de las piezas blancas, en formato "apellido, nombre".
6. Black: El jugador de las negras en el mismo formato.
7. Result: El resultado del juego. Sólo puede tener cuatro posibles valores: "1-0" (las blancas ganaron), "0-1" (Las negras ganaron), "1/2-1/2" (Tablas), o "*" (para otro, ejemplos: el juego está actualmente en disputa o un jugador falleció durante la partida).

Muchos otros pares de etiquetas son definidos por los estándares. Entre los que están:

- Time: La hora en que el juego empezó en formato "HH:MM:SS" de tiempo local.
- Termination: Da más detalles del fin del juego. Puede ser "abandono", "adjudicación" (resultado determinado por adjudicación de una tercera parte), "muerte", "emergencia", "normal", "infracción a las reglas", "tiempo acabado" o "no finalizado".
- FEN: La posición inicial del juego en notación de Forsyth-Edwards (del inglés Forsyth-Edwards Notation). Esta se utiliza para registrar juegos parciales, que empiezan en alguna posición determinada. También es necesaria para variantes del ajedrez como en el Ajedrez aleatorio de Fischer, donde la posición inicial

El texto de las jugadas describe los movimientos realizados en el juego. Esto debe incluir indicadores del número de la jugada (un número seguido de cero o más puntos) y notación estándar algebraica (NEA) [6].

El texto de las jugadas en NEA describe los movimientos realizados. En la mayoría de los casos, esto es simplemente la letra descriptiva de la pieza en inglés, una "x" si existe una captura, y el nombre algebraico de dos caracteres del escaque final a donde la pieza se desplazó. Las abreviaturas de las piezas en inglés son:

- K = (king) Rey
- Q = (queen) Dama
- R = (rook) Torre
- B = (bishop) Alfil
- N = (knight) Caballo
- P = (pawn) Peón

En NEA al peón se le da una abreviación vacía, pero en otros contextos su abreviación es "P". El nombre algebraico de cada escaque es el usual de la notación algebraica.

En algunos casos lo anterior puede ser ambiguo; si es así, el nombre algebraico de la pieza, su número de fila, o el escaque exacto es colocado después del nombre de la pieza que se mueve (en este orden de preferencia). De esta forma, "Nge2" se refiere a un caballo que está en g y se mueve a e2.

El enroque corto en NEA se indica con "O-O" y el largo con "O-O-O" (nótese que estas son letras os mayúsculas y no números ceros). Una promoción de un peón se denota añadiendo un signo "=" seguido del nombre algebraico de la pieza a la que el peón se promueve. Si el movimiento genera un jaque, se añade el signo "+"; si el movimiento implica un jaque mate se añade el signo de numeral "#".

Si el resultado del juego es cualquier otra cosa distinta a "*", el resultado se repite al final del texto de las jugadas.

Los comentarios deben ser añadidos ya sea con un ";" (un comentario de una sola línea) o con un "{" (que continuará hasta que aparece un "}". Los comentarios no se mezclan.

Ejemplo de un juego en formato PGN. Esta partida es muy famosa y se llama "La inmortal":

[Event "Informal Game"]
[Site "London, England ENG"]
[Date "1851.07.??"]

[Round "-"]
[White "Anderssen, Adolf"]
[Black "Kieseritzky, Lionel"]
[Result "1-0"]

1.e4 e5 2.f4 exf4 3.Bc4 Qh4+ 4.Kf1 b5 5.Bxb5 Nf6 6.Nf3 Qh6 7.d3 Nh5 8.Nh4 Qg5 9.Nf5 c6 10 g4 Nf6 11.Rg1 cxb5 12.h4 Qg6 13.h5 Qg5 14.Qf3 Ng8 15.Bxf4 Qf6 16.Nc3 Bc5 17.Nd5 Qxb2 18.Bd6 Bxg1 19.e5 Qxa1+ 20.Ke2 Na6 21.Nxg7+ Kd8 22.Qf6+ Nxf6 23.Be7# 1-0

Las variantes de ajedrez pueden también ser grabadas utilizando PGN. Estas son comúnmente anotadas con una etiqueta llamada "Variant" (variante) en donde se da como valor el nombre de las reglas. Sea cuidadoso de no utilizar el término "Variation" (Variación) que significa el nombre de una variación de una apertura. Nótese que muchos programas de ajedrez soportan al menos algunas variantes del ajedrez.

2.3 Herramientas de de visualización de partidas de ajedrez en formato PGN

Las herramientas en código libre disponibles para reproducir partidas de ajedrez en formato PGN pueden dividirse en dos grupos. A continuación se explicará cada una de estas tecnologías y se detallarán las principales herramientas implementadas en cada una de ellas:

Javascript:

JavaScript es, según la definición en wikipedia [7] un lenguaje de programación interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C. Al igual que Java, JavaScript es un lenguaje orientado a objetos propiamente dicho, ya que dispone de Herencia, si bien esta se realiza siguiendo el paradigma de programación basada en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad.

Todos los navegadores modernos interpretan el código JavaScript integrado dentro de las páginas web. De hecho JavaScript se ha convertido en el lenguaje de scripting por excelencia y, sin lugar a dudas, el más usado. JavaScript se ejecuta en el agente de usuario al mismo tiempo que las sentencias van descargándose junto con el código HTML. Las principales ventajas de esta tecnología son las siguientes:

- El lenguaje de scripting se ejecuta en un entorno controlado que impide su acceso a partes críticas del sistema cliente.

- El código Javascript se ejecuta en el cliente por lo que se reduce la carga computacional del servidor; un script ejecutado en el servidor, sin embargo,

sometería a éste a dura prueba y los servidores de capacidades más limitadas podrían resentir de una continua solicitud por un mayor número de usuarios.

Por el contrario, los principales inconvenientes son los siguientes:

- Los script tienen capacidades limitadas, por razones de seguridad, por lo cual no es posible hacer todo con Javascript, sino que es necesario usarlo conjuntamente con otros lenguajes evolucionados, posiblemente más seguros, como Java. Dicha limitación es aún más evidente si queremos operar en el hardware del ordenador.

- El código es visible y puede ser leído por cualquiera, incluso si está protegido con las leyes del copyright.

- El código del script debe descargarse completamente antes de poderse ejecutar. Por lo tanto, si los datos que un script utiliza son muchos, el tiempo que tardará en descargarse será muy largo, mientras que la consulta de la misma base de datos en el servidor sería más rápida.

Palview
Creador: Andrew Templeton.
Enlace: http://www.enpassant.dk/chess/palview/index.htm
Palview es una aplicación de escritorio que genera código html con javascript embebido a partir de un archivo en formato PGN. Tiene múltiples opciones de generación de este código, que resulta en distintos estilos de página. La interfaz gráfica es atrayente.

Figura 1.1 Fuente: http://www.enpassant.dk/chess/palview/p3demo/normal1.htm

pgn2html
Creador: Werner Mueller.
Enlace: http://www.muewer.homepage.t-online.de/pgn2html.html

Esta aplicación de escritorio genera el código javascript necesario para visionar una partida. Gráficamente es bastante atrayente y la carga es rápida. Soporta variaciones y se pueden usar flechas para mostrar un plan. Interfaz muy sencilla.

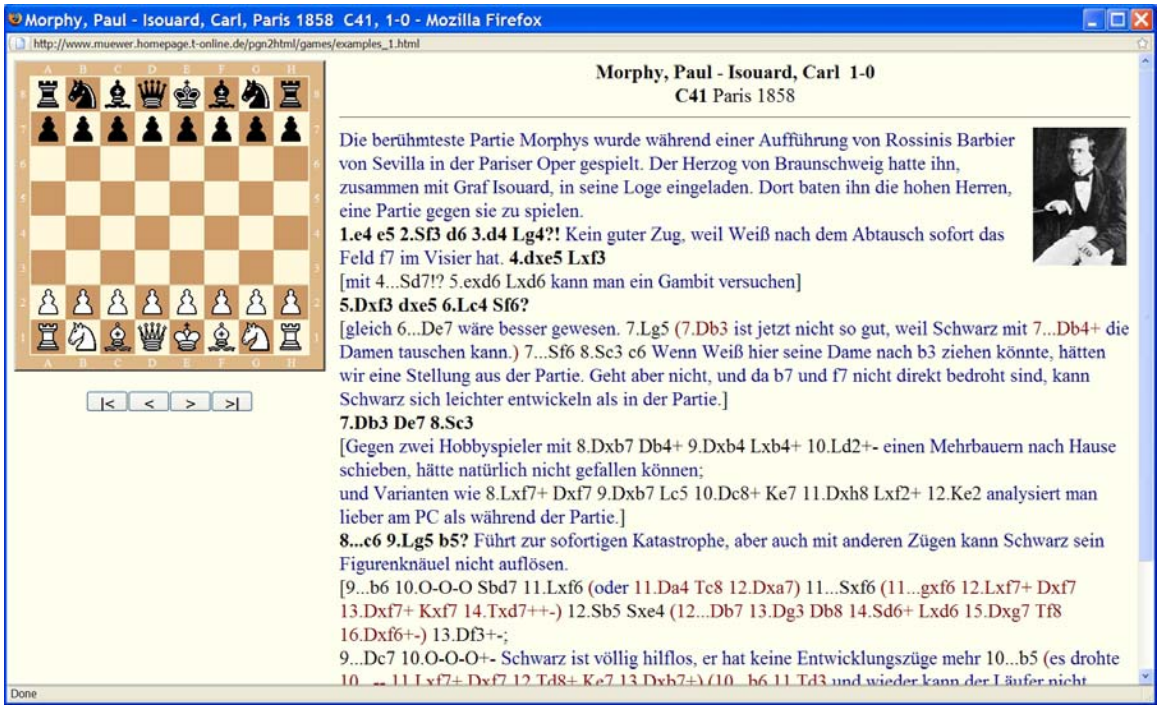


Figura 1.2 Fuente: http://www.muewer.homepage.t-online.de/pgn2html/games/examples_1.html

PGN to JS v3
Creador: Uwe Auerswald.
Enlace: http://www.mailchess.de/pgntojse.html
Al igual que Palview, genera código html con javascript embebido a partir de un archivo en formato PGN. Interfaz sencilla, muy similar al de **pgn2html**

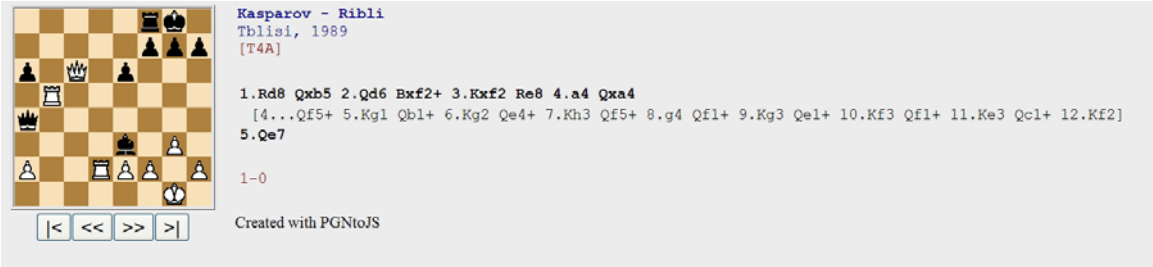


Figura 1.3 Fuente: captura del programa

LT-Pgn-Viewer
Creador: Lutz Tautenhahn.
Enlace: http://www.lutanho.net/pgn/pgnviewer.html
Visor de partidas en formato PGN para páginas web. El funcionamiento es mediante copia del texto en la interfaz de la página, lo que permite visualizar la partida. Una parte

interesante es que la interfaz permite realizar anotaciones en la partida, las cuales se muestran al reproducirla. El autor ofrece documentación para la integración de la herramienta en una página web. La interfaz contiene numerosas opciones, lo que complica el manejo de la aplicación sustancialmente. Sin embargo podrían reutilizarse los componentes que resultasen más interesantes.

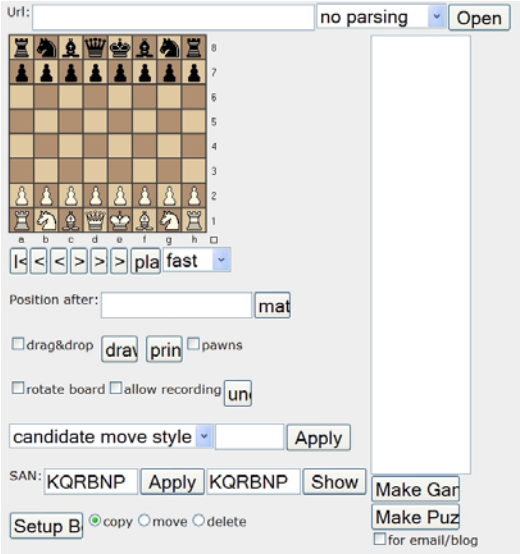


Figura 1.4 Fuente: http://www.lutanho.net/pgn/pgnviewer.html

Applets

Un applet [8] es un componente de una aplicación que se ejecuta en el contexto de otro programa, por ejemplo un navegador web. El applet debe ejecutarse en un contenedor, que lo proporciona un programa anfitrión, mediante un plugin, o en aplicaciones como teléfonos móviles que soportan el modelo de programación por applets.

A diferencia de un programa, un applet no puede ejecutarse de manera independiente, ofrece información gráfica y a veces interactúa con el usuario, típicamente carece de sesión y tiene privilegios de seguridad restringidos. Un applet normalmente lleva a cabo una función muy específica que carece de uso independiente.

Los applets de Java pueden correr en un navegador web utilizando la Java Virtual Machine (JVM), o en el AppletViewer de Sun. El Navegador que carga y ejecuta el applet se conoce en términos genéricos como el contenedor de Applets. El kit de desarrollo de Software para Java 2 (J2SDK) 1.6 (también conocida como 6.0) incluye el contenedor de Applets, llamado appletviewer, para probar los applets antes de incrustarlos en una página Web.

Entre sus características podemos mencionar un esquema de seguridad que permite que los applets que se ejecutan en el equipo no tengan acceso a partes sensibles (por ej. no pueden escribir archivos), a menos que uno mismo le dé los permisos necesarios en el sistema; la desventaja de este enfoque es que la entrega de permisos es engorrosa para el

usuario común, lo cual juega en contra de uno de los objetivos de los Java applets: proporcionar una forma fácil de ejecutar aplicaciones desde el navegador web.

En Java un applet (Subprograma), es un programa que puede incrustarse en un documento HTML; es decir en una página Web. Cuando un Navegador carga una página Web que contiene un Applet, éste se descarga en el navegador Web y comienza a ejecutarse. Esto nos permite crear programas que cualquier usuario puede ejecutar con tan solo cargar la página Web en su navegador. Este es el interés, proporciona una forma a través de la cual se puede distribuir software al cliente desde el servidor, en el momento en que el cliente necesite ese software, y no antes, con lo cual siempre tendrá el cliente la última versión de ese software, se actualice cuando se actualice. Además, tal como está diseñado Java, el programador necesita crear su programa una sola vez, y ya estará listo para ser ejecutado en todas las plataformas que dispongan de un navegador con soporte Java.

Con Java se podrá realizar tanto trabajo como sea posible en el cliente antes y después de hacer peticiones al servidor. Por ejemplo, se puede evitar el enviar una petición a través de Internet mientras el usuario no haya introducido los parámetros correctos de esa petición, que estará chequeando el cliente, sin necesidad de tener que consultar continuamente al servidor; con ello se gana en velocidad de respuesta ante el usuario, una reducción general en el tráfico de red y una gran descarga de trabajo para el servidor.

Una de las ventajas de los applets sobre los scripts es que están en forma compilada, con lo cual el código fuente no es visible, y aunque pueda ser descompilado por determinadas herramientas, no está del todo accesible. Además, un applet puede comprimir varios módulos, utilizando ficheros JAR (Java Archive), evitando múltiples conexiones con el servidor, ya que en una sola se descargan todos los componentes necesarios.

Los applets de Java suelen tener las siguientes ventajas:

- Son multiplataforma (funcionan en Linux, Windows, Mac OS, y en cualquier sistema operativo para el cual exista una JVM).
- El mismo applet puede trabajar en "todas" las versiones de Java, y no sólo la última versión del plug-in. Sin embargo, si un applet requiere una versión posterior de la JRE (Java Runtime Environment), el cliente se verá obligado a esperar durante la descarga de la nueva JRE.
- Es soportado por la mayoría de los navegadores Web.
- Puede ser almacenado en la memoria cache de la mayoría de los navegadores Web, de modo que se cargará rápidamente cuando se vuelva a cargar la página Web, aunque puede quedar atascado en la caché, causando problemas cuando se liberan nuevas versiones.
- Puede tener acceso completo a la máquina en la que se está ejecutando, si el usuario lo permite.
- Puede ejecutarse con velocidades comparables (pero en general más lento) a la de otros lenguajes compilados, como C + +, pero muchas veces más rápida que la de JavaScript. Cabe indicar que recientemente se están desarrollando intérpretes de

- Puede trasladar el trabajo del servidor al cliente, haciendo una solución Web más escalable tomando en cuenta el número de usuarios / clientes.

Los applets de Java suelen tener las siguientes desventajas:

- Requiere el plug-in de Java, que no está disponible por defecto en todos los navegadores web.
- Sun no ha creado una implementación del plug-in para los procesadores de 64 bits.
- No puede iniciar la ejecución hasta que la JVM esté en funcionamiento, y esto puede tomar tiempo la primera vez que se ejecuta un applet.
- Si no está firmado como confiable, tiene un acceso limitado al sistema del usuario - en particular no tiene acceso directo al disco duro del cliente o al portapapeles.
- Algunas organizaciones sólo permiten la instalación de software a los administradores. Como resultado, muchos usuarios (sin privilegios para instalar el plug-in en su navegador) no pueden ver los applets.
- Un Applet podría exigir una versión específica del JRE.

Pgn Reader
Creador: Kevin Coulombe.
Enlace: http://www.stonkie.com/en/pgnreader/index.html
Es un applet Java sencillo, rápido y amigable para visionar partidas de ajedrez. El autor proporciona documentación detatallada sobre cómo implementar el applet en un página web.

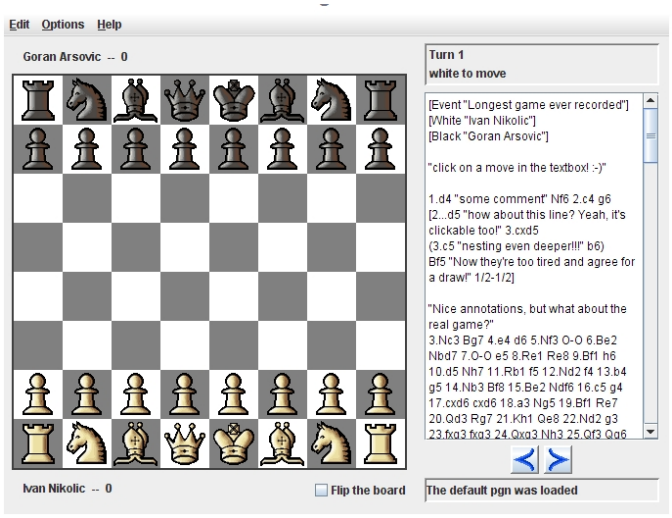


Figura 1.5 Fuente: http://www.stonkie.com/en/pgnreader/v2.2.html

Chess TutorTM
Creador: Eduardo Suastegui.
Enlace: http://www.marochess.de/ChessTutor/
Lee archivos PGN y permite el visualizado interactivo de partidas. El tiempo de carga es superior al de otros applets, ya que tiene mayor número de opciones. Puede trabajar con archivos que contengan varias partidas, leer archivos PGZ (Archivos PGN con compresión ZIP), manejar varios archivos a la vez y consultar los archivos de juego por cambios, lo que es muy útil para partidas en directo. El autor proporciona el código fuente.

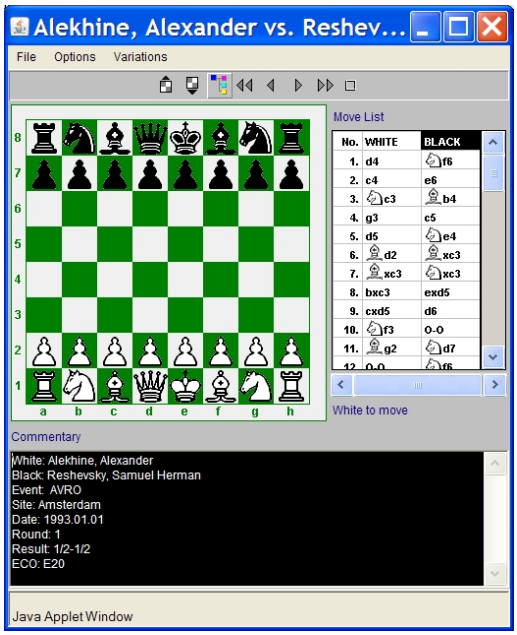


Figura 1.6 Fuente: http://www.marochess.de/ChessTutor/

Chess Viewer
Creador: Andrew Gove.
Enlace: http://www.chessclub.com/chessviewer/
Permite visualizar de forma interactiva una partida. La carga es muy rápida. Un gran inconveniente es que no lee directamente archivos PGN, sino documentos HTML. La página proporciona un traductor de archivos PGN al formato HTML que usa la herramienta.

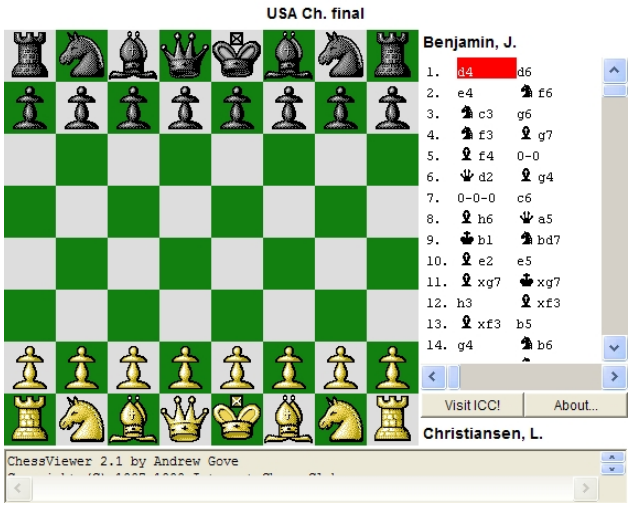


Figura 1.7 Fuente: http://www.chessclub.com/chessviewer/example.html

Misty Beach PGN viewer
Creador: Mark Roulo.
Enlace: http://www.mistybeach.com/
Lee archivos PGN y permite visualizar la partida. Su principal ventaja es la rapidez de carga. Y su inconveniente es que gráficamente no es de los más atractivos.

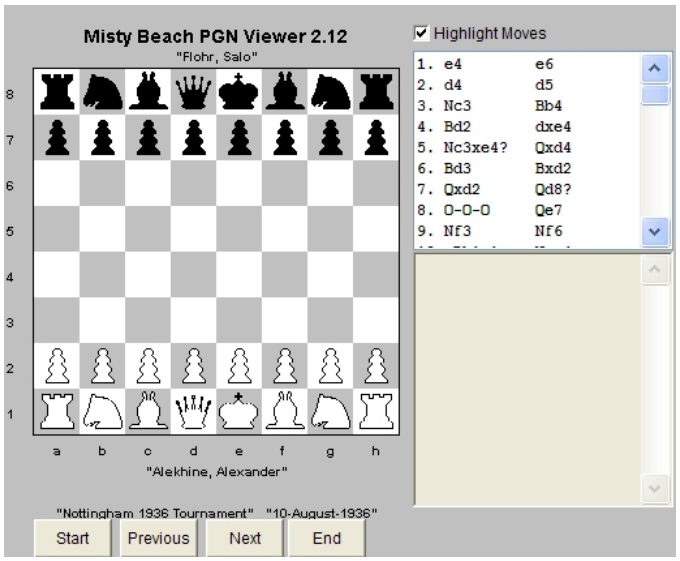


Figura 1.8 Fuente: http://www.mistybeach.com/products/PGNViewer/Html/Nottingham1936Game001.html

Montreux
Creador: JP Hendriks.
Enlace: http://www.jbfsoftware.com/

Visor para archivos PGN. Su principal ventaja es la rapidez de carga. El autor proporciona también instrucciones para su integración en una página web. Su principal desventaja es que gráficamente es poco atrayente.

Figura 1.9 Fuente: http://www.jbfsoftware.com/

MyChessViewer
Creador: Michael Keating.
Enlace: http://www.mychess.com/
Un applet gráficamente atrayente para la visualización de archivos PGN. Incluye opciones para varias partidas, fragmentos de partidas, comentarios y problemas. Una de sus mayores ventajas es su tamaño y la velocidad de carga. También puede leer archivos PGN en formato comprimido. El autor proporciona el código fuente.

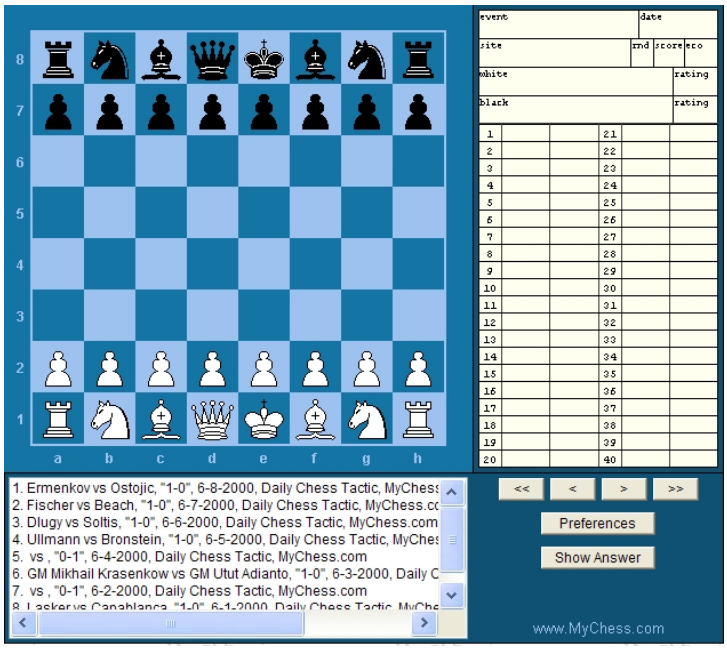


Figura 1.10 Fuente: http://www.mychess.com/appletviewer/Polgar-Anand.htm

Sjkbase
Creador: Odd Gunnar Malin.
Enlace: http://home.online.no/~malin/sjakk/download/sjkbase11/installation.html
Hay dos versiones de este applet, una para la reproducción de partidas en formato PGN y otra para resolver problemas de ajedrez. La aplicación es muy pequeña y de carga rápida. Gráficamente es atrayente, lee posiciones EPD/FEN y permite seleccionar entre varios idiomas. El código fuente también está disponible.

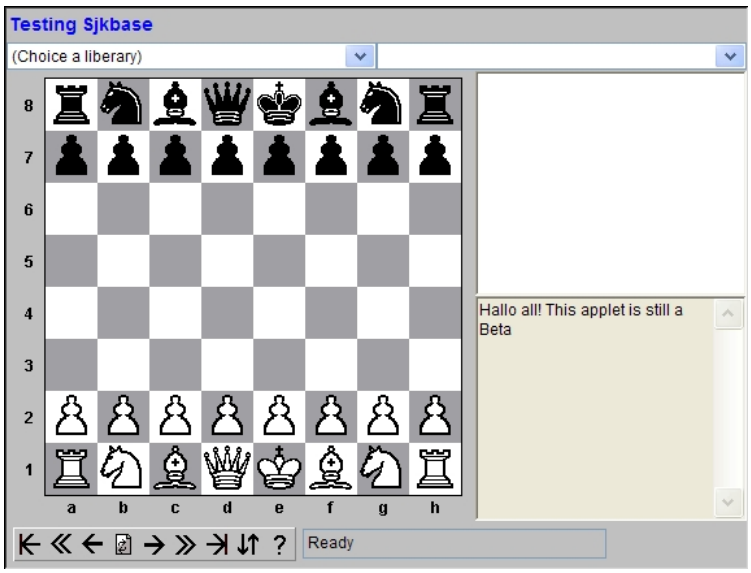


Figura 1.11 Fuente: http://home.online.no/~malin/sjakk/download/sjkbase11/sjkbase.html

2.4 Bases de datos de partidas

Hay varias Bases de Datos de partidas de ajedrez. Muchas de ellas no son libres, y sólo es posible acceder a ellas mediante pago. Sin embargo existen algunas bases interesantes de contenido libre. Estas son las más importantes.

Scid
http://scid.sourceforge.net/
Es una aplicación que permite acceder a datos de partidas de ajedrez, editarlas y realizar su búsqueda atendiendo a varios criterios.
Scid usa un formato de base de datos particular, basado en tres archivos, el cual es compacto y de elevada rapidez. Puede convertir y leer al formato PGN. El mayor impedimento es que no admite la posibilidad de editar los archivos PGN, ya que sólo permite abrirlos como archivos de sólo lectura. Además, abrir estos archivos consume gran cantidad de memoria y son lentos de cargar, por lo que para archivos PGN grandes se recomienda pasarlos al formato de la base de datos scid.

José
http://jose-chess.sourceforge.net/
José es una herramienta gráfica para visualización y almacenamiento de partidas de ajedrez. Lo más interesante de esta herramientas es que la base de datos en que se basa es MySQL, la más importante de código libre. Incorpora una herramienta para la utilización de la base de datos en servidores web, pero está muy poco documentada y es poco flexible.

ChessX
http://chessx.sourceforge.net/
Es una base de datos de código libre multiplataforma. Su interfaz permite leer archivos PGN, trabajar con distintas bases de datos simultáneamente y una base personalizada para el usuario con soporte para imágenes. Está en desarrollo y el principal de los próximos objetivos es guardar en la base directamente los archivos PGN.

ChessDB
http://chessdb.sourceforge.net/
ChessDB está basada en Scid, que fue la principal base ajedrecística libre. ChessDB tiene un gran número de características. Algunas son bastante interesantes, como la posibilidad de salvar partidas ya sea en el formato estándar PGN, en HTML o LaTeX, o anotar las partidas agregando comentarios de texto, símbolos estándar, señales coloreadas a las partidas o variantes mostrando diferentes líneas que pudieron ser de interés.

Colecciones de partidas
Hay numerosos sitios que proporciona partidas en formato PGN. En estos sitios se pueden descargar ficheros de texto con un gran número de partidas. Es interesante para generar el contenido de una base de datos propia con archivos en formato PGN. Para ello necesitaríamos, naturalmente, utilizar un Sistema de Gestión de Base de Datos. Estos son algunas de las colecciones más extensas:

- ***Twic:*** http://www.chesscenter.com/twic/twic.html
- ***University of Pittsburg:*** ftp://ftp.pitt.edu/group/chess
- ***Britbase*** http://www.saund.co.uk/britbase/
- ***Chessopeningspgn*** http://www.chessopeningspgn.com/chess/Openings.html
- ***SupremeChess*** http://www.supreme-chess.com/chess-game-collection-a.html
- ***Masterchessgames*** http://www.masterchessgames.com/
- ***chess-database*** http://www.chess-database.com/

2.5 Sistemas de Gestión de Bases de Datos

Hay una gran cantidad de Sistemas de Gestión. Los más importantes, por ser los más ampliamente utilizados son los siguientes:

MySQL
Es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones [9] [10]. MySQL AB —desde enero de 2008 una subsidiaria de Sun Microsystems y ésta a su vez de Oracle Corporation desde abril de 2009— desarrolla MySQL como software libre en un esquema de licenciamiento dual.

Por un lado se ofrece bajo la GNU GPL (GNU General Public License) para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en ANSI C.

Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y el copyright del código está en poder del autor individual, MySQL es propietario y está patrocinado por una empresa privada, que posee el copyright de la mayor parte del código. Esto es lo que posibilita el esquema de licenciamiento anteriormente mencionado.

MySQL es una base de datos muy rápida en la lectura cuando utiliza el motor no transaccional MyISAM, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. En aplicaciones web hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones.

Inicialmente, MySQL carecía de elementos considerados esenciales en las bases de datos relacionales, tales como integridad referencial y transacciones. A pesar de ello, atrajo a los desarrolladores de páginas web con contenido dinámico, justamente por su simplicidad. Pero poco a poco los elementos de los que carecía MySQL están siendo incorporados tanto por desarrollos internos, como por desarrolladores de software libre.

Algunas de sus características principales son las siguientes:

- Soporta gran cantidad de datos.
- Amplio subconjunto del lenguaje SQL. Algunas extensiones son incluidas igualmente.
- Disponibilidad en gran cantidad de plataformas y sistemas.
- Diferentes opciones de almacenamiento según si se desea velocidad en las operaciones o el mayor número de operaciones disponibles.
- Transacciones y claves foráneas.
- Conectividad segura.
- Replicación.
- Búsqueda e indexación de campos de texto.

PostgreSql
Es un sistema de gestión de base de datos relacional orientada a objetos de software libre, publicado bajo la licencia BSD [11].

Como muchos otros proyectos open source, el desarrollo de PostgreSQL no es manejado por una sola compañía sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales que trabajan en su desarrollo. Dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group).

Algunas de sus principales características son, entre otras:

- Alta concurrencia: mediante un sistema denominado MVCC (Acceso concurrente multiversión, por sus siglas en inglés) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo commit. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos.

- Amplia variedad de tipos nativos: PostgreSQL provee nativamente soporte para:

    - Números de precisión arbitraria.
    - Texto de largo ilimitado.
    - Figuras geométricas (con una variedad de funciones asociadas)
    - Direcciones IP (Internet Protocol, IPv4 e IPv6).
    - Bloques de direcciones estilo CIDR.
    - Direcciones MAC.
    - Arrays.

    Adicionalmente los usuarios pueden crear sus propios tipos de datos, que pueden ser por completo indexables gracias a la infraestructura GiST de PostgreSQL. Algunos ejemplos son los tipos de datos GIS creados por el proyecto PostGIS.

- Claves ajenas también denominadas Llaves ajenas o Claves Foráneas (foreign keys).
- Disparadores (triggers): Un disparador o trigger se define en una acción especifica basada en algo ocurrente dentro de la base de datos. En PostgreSQL esto significa la

- Vistas.
- Integridad transaccional.
- Herencia de tablas.
- Tipos de datos y operaciones geométricas.
- Funciones: bloques de código que se ejecutan en el servidor. Pueden ser escritos en varios lenguajes, con la potencia que cada uno de ellos da, desde las operaciones básicas de programación, tales como bifurcaciones y bucles, hasta las complejidades de la programación orientada a objetos o la programación funcional.

Oracle
Es un sistema de gestión de base de datos relacional, desarrollado por Oracle Corporation [12]. Se considera a Oracle como uno de los sistemas de bases de datos más completos, destacando las siguientes cualidades:

- Soporte de transacciones.
- Estabilidad.
- Escalabilidad.
- Soporte multiplataforma.

Ha sido criticada por algunos especialistas la seguridad de la plataforma, y las políticas de suministro de parches de seguridad, modificadas a comienzos de 2005 y que incrementan el nivel de exposición de los usuarios. En los parches de actualización provistos durante el primer semestre de 2005 fueron corregidas 22 vulnerabilidades públicamente conocidas, algunas de ellas con una antigüedad de más de 2 años.

Aunque su dominio en el mercado de servidores empresariales ha sido casi total hasta hace poco, recientemente sufre la competencia del Microsoft SQL Server de Microsoft y de la oferta de otros RDBMS con licencia libre como PostgreSQL, MySql o Firebird. Las últimas versiones de Oracle han sido certificadas para poder trabajar bajo GNU/Linux.

La tecnología Oracle se encuentra prácticamente en todas las industrias alrededor del mundo y en las oficinas de 98 de las 100 empresas Fortune 100. Oracle es la primera compañía de software que desarrolla e implementa software para empresas 100 por ciento activado por Internet a través de toda su línea de productos: base de datos, aplicaciones comerciales y herramientas de desarrollo de aplicaciones y soporte de decisiones. Oracle es el proveedor mundial líder de software para administración de información, y la segunda empresa de software.

Microsoft SQL Server
Sistema de gestión de bases de datos relacionales basado en el lenguaje Transact-SQL, y específicamente en Sybase IQ, capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea [13]. Sus principales ventajas son las que se citan a continuación:

- Soporte de transacciones.
- Escalabilidad, estabilidad y seguridad.
- Soporta procedimientos almacenados.
- Incluye también un potente entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente.
- Permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y las terminales o clientes de la red sólo acceden a la información.
- Además permite administrar información de otros servidores de datos.

Para el desarrollo de aplicaciones más complejas (tres o más capas), Microsoft SQL Server incluye interfaces de acceso para varias plataformas de desarrollo, entre ellas .NET, pero el servidor sólo está disponible para Sistemas Operativos Windows.

Adaptive Server Enterprise

También conocido como ASE, es el motor de bases de datos (RDBMS) insignia de la compañía Sybase [14]. ASE es un sistema de gestión de datos, altamente escalable, de alto rendimiento, con soporte a grandes volúmenes de datos, transacciones y usuarios, y de bajo costo, que permite:

- Almacenar datos de manera segura
- Tener acceso y procesar datos de manera inteligente
- Movilizar datos

ASE fue liberado en 1999, brindando soporte para Java en la base de datos, alta disponibilidad y gestión de transacciones distribuidas. En 2001, ASE 12.5 fue lanzada, con características tales como asignación dinámica de memoria, soporte para XML en la base de datos y conexiones seguras con SSL (Secure Sockets Layer), entre otros. Algunas de sus principales características son las siguientes:

- Administrador lógico de recursos y tareas
- Múltiples esquemas de bloqueo de datos
- Copias de respaldo en línea y de alto rendimiento
- Integración transparente con orígenes de datos remotos
- Programador de tareas
- Conexiones seguras con SSL
- Soporte a LDAP para autenticación de usuarios y conectividad cliente/servidor
- Soporte a múltiples herramientas de desarrollo y lenguajes de programación, como PowerBuilder, Visual Basic, Java, C, PHP, etc.
- Soporte a múltiples protocolos de conectividad, como Open Client (propio de Sybase), ODBC, OLE DB, ADO.NET y JDBC.

Interbase
Es un Sistema de Administración de Base de Datos Relacionales que corre en plataformas Linux, Microsoft Windows y Solaris [15]. Sus principales características son las que se citan a continuación:

- Bajo consumo de recursos: una instalación completa del servidor de Interbase 7 requiere aproximadamente 40Mb en disco. Esto es significativamente mas pequeño que la instalación del cliente de muchos servidores de base de datos de otras compañías. El servidor usa muy poca memoria mientras está ocioso. Una instalación mínima de un cliente InterBase requiere aproximadamente 400Kb de espacio en disco.
- Administración mínima: los servidores Interbase normalmente no requieren de administradores a tiempo completo.
- Control de concurrencia
- Recuperación: la mayoría de los sistemas utilizan logs para realizar esta operación, lo que puede tomar mucho tiempo e incluso necesitar de intervención manual. En cambio, la recuperación en Interbase es casi instantánea y nunca falla.

Entre sus principales desventajas destaca el hecho de que ciertas operaciones son más difíciles de implementar debido a su arquitectura multi-generacional, y por lo tanto se ejecutan más lentas en comparación a otras implementaciones tradicionales.

Firebird
Es un sistema de administración de base de datos relacional de código abierto, basado en la versión 6 de Interbase [16]. Su código fue reescrito de C a C++. Éstas son sus características más destacables:

- Es multiplataforma, y actualmente puede ejecutarse en los sistemas operativos: Linux, HP-UX, FreeBSD, Mac OS, Solaris y Microsoft Windows.
- Ejecutable pequeño, con requerimientos de hardware bajos.
- Soporte de transacciones ACID y claves foráneas.
- Es medianamente escalable.
- Buena seguridad basada en usuarios/roles.
- Diferentes arquitecturas, entre ellas el Firebird incrustado (embedded server) que permite ejecutar aplicaciones monousuario en ordenadores sin instalar el software Firebird o la arquitectura Cliente/Servidor sobre protocolo TCP/IP.
- Bases de datos de sólo lectura, para aplicaciones que corran desde dispositivos sin capacidad de escritura, como cd-roms.
- Existencia de controladores ODBC, OLEDB, JDBC, PHP, Perl, .net, etc.
- Requisitos de administración bajos, siendo considerada como una base de datos libre de mantenimiento, al margen de la realización de copias de seguridad.
- Pleno soporte del estándar SQL-92, tanto de sintaxis como de tipos de datos.
- Completo lenguaje para la escritura de disparadores y procedimientos almacenados denominado PSQL.
- Capacidad de almacenar elementos BLOB (Binary Large OBjects).

- Soporte de User-Defined Functions (UDFs).
- Versión autoejecutable, sin instalación, excelente para la creación de catálogos en CD-Rom y para crear versiones de evaluación de algunas aplicaciones.

2.6 Bibliotecas gráficas

Las principales bibliotecas gráficas de Java son AWT y Swing. Sus principales características son las siguientes:

AWT
La Abstract Window Toolkit es un kit de herramientas de gráficos, interfaz de usuario, y sistema de ventanas independiente de la plataforma original de Java [19]. AWT es ahora parte de las Java Foundation Classes (JFC) - la API estándar para suministrar una interfaz gráfica de usuario (GUI) para un programa Java.

AWT suministra un nivel de abstracción muy fino sobre la interfaz de usuario nativa subyacente. Algunos desarrolladores de aplicaciones prefieren este modelo porque suministra un alto grado de fidelidad al kit de herramientas nativo subyacente y mejor integración con las aplicaciones nativas. En otras palabras, un programa GUI escrito usando AWT parece como una aplicación nativa Microsoft Windows cuando se ejecuta en Windows, pero el mismo programa parece una aplicación nativa Apple Macintosh cuando se ejecuta en un Mac.

AWT continúa suministrando el núcleo del subsistema de eventos GUI y la interfaz entre el sistema de ventanas nativo y la aplicación Java, suministrando la estructura que necesita Swing. También suministra gestores de disposición básicos, un paquete de transferencia de datos para uso con el Bloc de notas y Arrastrar y Soltar, y la interfaz para los dispositivos de entrada tales como el ratón y el teclado.

Swing
Es una biblioteca gráfica para Java [18] [19]. Incluye widgets para interfaz gráfica de usuario tales como cajas de texto, botones, desplegables y tablas.

Desde sus inicios el entorno Java ya contaba con una biblioteca de componentes gráficos conocida como AWT. Esta biblioteca estaba concebida como una API estandarizada que permitía utilizar los componentes nativos de cada sistema operativo. Swing introdujo un mecanismo que permitía que el aspecto de cada componente de una aplicación pudiese cambiar sin introducir cambios sustanciales en el código de la aplicación. La introducción de soporte ensamblable para el aspecto permitió a Swing emular la apariencia de los componentes nativos manteniendo las ventajas de la independencia de la plataforma. También contiene un conjunto de herramientas que nos permiten crear una interfaz atractiva para los usuarios.

Swing es por tanto, una plataforma independiente, un framework Modelo Vista Controlador para Java. Sigue un simple modelo de programación por hilos, y posee las siguientes características principales:

- Independencia de plataforma.
- Extensibilidad: es una arquitectura altamente particionada: los usuarios pueden proveer sus propias implementaciones modificadas para sobrescribir las implementaciones por defecto. Se puede extender clases existentes proveyendo alternativas de implementación para elementos esenciales.
- Personalizable: dado el modelo de representación programático del framework de swing, el control permite representar diferentes aspectos ("look and feel") (desde aspecto MacOS hasta aspecto Windows XP). Además, los usuarios pueden proveer su propia implementación de aspecto, que permitirá cambios uniformes en el aspecto existente en las aplicaciones Swing sin efectuar ningún cambio al código de aplicación.

Sus ventajas más significativas son las siguientes:

- El diseño en Java puro posee menos limitaciones de plataforma.
- El desarrollo de componentes Swing es más activo.
- Los componentes de Swing soportan más características.

2.7 Licencias de software

Una licencia de software es un contrato entre el licenciante (autor/titular de los derechos de explotación/distribuidor) y el licenciatario del programa informático (usuario consumidor /usuario profesional o empresa), para utilizar el software cumpliendo una serie de términos y condiciones establecidas dentro de sus cláusulas, según la definición de wikipedia [20].

Las licencias de software pueden establecer entre otras cosas: la cesión de determinados derechos del propietario al usuario final sobre una o varias copias del programa informático, los límites en la responsabilidad por fallos, el plazo de cesión de los derechos, el ámbito geográfico de validez del contrato e incluso pueden establecer determinados compromisos del usuario final hacia el propietario, tales como la no cesión del programa a terceros o la no reinstalación del programa en equipos distintos al que se instaló originalmente.

Según los derechos que cada autor se reserva sobre su obra, las licencias de software se pueden clasificar de la siguiente forma:

- Licencia de software libre permisiva: se puede crear una obra derivada sin que ésta tenga obligación de protección alguna. Muchas licencias pertenecen a esta clase, entre otras:

        - Academic Free License v.1.2.
        - Apache Software License v.1.1.
        - Artistic License v.2.0
        - Attribution Assurance license.
        - BSD License.
        - MIT License.
        - University of Illinois/NCSA Open Source License.
        - W3C Software Notice and License.
        - Zope Public License v.2.0
        - Open LDAP License v.2.7
        - Perl License.
        - Academic Free License v.3.0
        - Python License v.2.1
        - PHP License v.3.0
        - Q Public License v.1.0

- Licencia de software libre robusta: estas licencias aplican algunas restricciones a las obras derivadas, haciendo que según el grado de aplicación se puedan dividir a su vez en dos subcategorias:

    - Licencias de software libre robustas fuertes: las licencias de software libre robustas fuertes o con copyleft fuerte, contienen una cláusula que obliga a que las obras derivadas o modificaciones que se realicen al software original se deban licenciar bajo los mismos términos y condiciones de la licencia original. Entre las licencias de esta categoría están:

        - Common Public License v.1.0.
        - GNU General Public License v.2.0.
        - GNU General Public License v.3.0.
        - Eclipse Public License.
        - eCos License v.2.0
        - Sleepycat Software Product License.
        - Affero License v.1.0
        - Affero License v.2.0
        - OpenSSL License.

    - Licencias de software libre robustas débiles: las licencias de software libre robustas débiles, con copyleft débil/suave o híbridas, contienen una cláusula que obliga a que las modificaciones que se realicen al software original se deban licenciar bajo los mismos términos y condiciones de la licencia original, pero que las obras derivadas que se puedan realizar de él puedan ser licenciadas bajo otros términos y condiciones distintas. Entre las licencias de esta categoría están:

        - GNU Lesser General Public License v.2.1.
        - Mozilla Public License

    - Open Source License.
    - Apple Source License v.2.0
    - CDDL.
    - EUPL.

- Licencia de software no libre: estas licencias también se conocen con el nombre de software privativo. En ellas los propietarios establecen los derechos de uso, distribución, redistribución, copia, modificación, cesión y en general cualquier otra consideración que se estime necesaria. Este tipo de licencias, por lo general, no permiten que el software sea modificado, desensamblado, copiado o distribuido de formas no especificadas en la propia licencia (piratería de software), regula el número de copias que pueden ser instaladas e incluso los fines concretos para los cuales puede ser utilizado. La mayoría de estas licencias limitan fuertemente la responsabilidad derivada de fallos en el programa.

- Software de dominio público (sin licencia): se permite uso, copia, modificación o redistribución con o sin fines de lucro.

Capítulo 3

Requisitos

En los capítulos previos ya se ha esbozado levemente alguno de los requisitos funcionales principales de la aplicación. De una forma estructurada se procederá a una exposición detallada de los requisitos tanto funcionales como no funcionales de la aplicación:

3.1 Base de datos

Requisitos funcionales

- La base de datos debe permitir el almacenamiento de partidas de ajedrez.
- La base de datos debe permitir también el almacenamiento de información relativa al usuario.
- Cada usuario deber poder ser identificado unívocamente.
- La base de datos debe permitir el almacenamiento de la información introducida por los usuarios a modo de etiquetas sobre una partida.
- Las etiquetas, o anotaciones, deberán poder ser asociadas a los siguientes fragmentos dentro de una partida: movimiento, posición o partida completa.
- Debe permitirse asociar una anotación a un rango de movimientos o posiciones.
- Debe garantizarse que cada anotación esté asociada a un solo usuario y a una sola partida.
- Debe ofrecerse un medio de importar en la base de datos partidas de ajedrez obtenidas de otras fuentes.

Requisitos NO funcionales

- Debe garantizarse que no existan partidas duplicadas bajo nombres distintos en la base de datos.
- Debe garantizarse que en ciertos campos el valor que se introduzca en la base de datos sea restringido a ciertos valores.
- El formato de las partidas con que trabajará la base de datos debe ser PGN.
- Cada archivo PGN cargado en el sistema contendrá una única partida.
- El tiempo de respuesta a una consulta a la base de datos debe ser reducido.
- El sistema de gestión de bases de datos debe ser software de licencia libre.
- El sistema de gestión de bases de datos debe posibilitar el acceso múltiple a la base de datos.
- El sistema de gestión de bases de datos debe posibilitar también la introducción de una cantidad grande de datos.

3.2 Herramienta de visualización de partidas

Requisitos funcionales

- Un usuario debe poder efectuar una búsqueda de las partidas almacenadas en la base de datos.
- La búsqueda deber poder efectuarse en función de parámetros básicos que identifiquen de manera clara una partida.
- El usuario debe poder visualizar el resultado de la búsqueda de partidas en la forma de una lista en la que se identifiquen de algún modo las distintas partidas.
- También debe poder seleccionar una partida entre el resultado de la búsqueda.
- La herramienta debe permitir reproducir la partida de forma visual.
- La herramienta debe también permitir el visualizado de las anotaciones asociadas a la partida seleccionada introducidas tanto por el propio usuario como por otros usuarios.
- Debe permitirse la introducción de anotaciones en la partida la partida actualmente visualizada.
- Las anotaciones deben poder realizarse sobre los siguientes fragmentos dentro de una partida: movimiento, posición o partida completa.
- Debe permitirse asociar una anotación a un rango de movimientos o posiciones.

Requisitos NO funcionales

- Debe impedirse la introducción por parte del usuario de datos inconsistentes en los rangos de movimientos y posiciones.
- Debe evitarse el retorno de partidas duplicadas como resultado de la búsqueda.
- Debe evitarse en la medida de lo posible inconsistencia en los parámetros de búsqueda.
- Debe garantizarse un tiempo de recuperación de partidas en la búsqueda razonable.
- Debe garantizarse un tiempo de recuperación de una partida específica reducido.
- Debe garantizarse que el tiempo de introducción de una anotación también sea reducido.
- Deberá comprobarse que la herramienta pueda ser desplegada en los principales navegadores.
- El tiempo de carga de la aplicación por el navegador no debe ser excesivo.
- Debe procurarse que la interfaz sea sencilla, amigable e intuitiva.

3.3 Herramienta de gestión de usuarios

Requisitos funcionales

- Debe permitirse la creación de una nueva cuenta de usuario.
- Debe asegurarse de que el nuevo usuario pueda ser identificado unívocamente.

- Se debe habilitar la operación de autenticación que permita acceder a la herramienta de visualización de partidas.
- Debe comprobarse que sólo usuarios registrados puedan acceder a la aplicación.
- Debe informarse al usuario cuando queden campos obligatorios sin responder.

Requisitos NO funcionales

- Debe procurarse que la interfaz sea sencilla, amigable e intuitiva.
- Deberá comprobarse que la herramienta pueda ser desplegada en los principales navegadores.

Capítulo 4

Arquitectura de la aplicación

En el capítulo anterior se ha dividido la lista de requisitos en una serie de bloques diferenciados entre sí por el tipo de operaciones que se realizarán en ellos. Estos bloques nos proporcionan una estructura inicial sobre la que basar el diseño. Por lo tanto, la aplicación en que se ocupa este proyecto se estructurará mediante cuatro módulos diferenciados:

- Módulo 1, gestión de usuarios: constará de una serie de páginas JSP que permitirán a un usuario abrir una cuenta y autenticarse. La información acerca de los usuarios del sistema se almacena en la base de datos.

- Módulo 2, herramienta de visualización de partidas: *applet* Java que permite visualizar partidas de ajedrez y realizar anotaciones sobre ellas. El applet se conecta a su vez a la base de datos mediante un servlet.

- Módulo 3, base de datos: base de datos que contendrá la información relativa a las partidas.

- Módulo 4, herramienta pobladora de base de datos: aplicación que permite importar partidas procedentes de otras fuentes en la base de datos.

La siguiente figura muestra un esquema de los distintos módulos y las relaciones entre los mismos. En los siguientes apartados se procederá a una descripción más detallada de cada uno de los módulos esbozados y de sus interrelaciones.

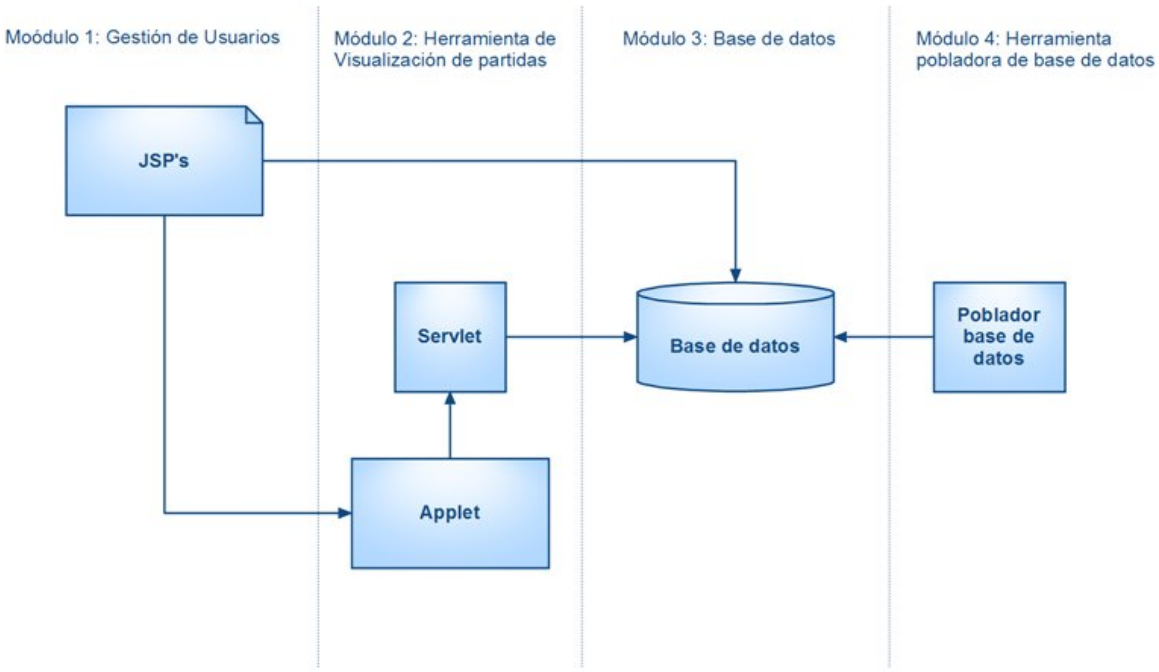


Figura 4.1.1 Diagrama modular de los distintos componentes de la aplicación

4.1 Base de datos

La información relativa a las partidas deberá almacenarse en una base de datos, como ya se ha apuntado. Se han considerado las bases de datos específicas de partidas de ajedrez ya existentes y la mayoría de ellas convierten los archivos PGN a un formato propio, en lugar de almacenar los archivos íntegramente, lo que provoca una dependencia a dicho formato. Pero dado que se necesita almacenar información adicional de anotaciones, se considera más adecuado diseñar un formato específico de almacenamiento en la base de datos. Esto también ahorraría la conversión de PGN al formato específico de cada base de datos y de éste a datos que pueda procesar la herramienta de visualización.

En la base de datos se almacenará, por tanto, toda la información relativa a las partidas de ajedrez, esto es, información básica de las partidas, anotaciones realizadas sobre ellas y la ruta a los ficheros PGN que contienen la partida. Estos ficheros serán almacenados en el servidor y contendrán cada uno una única partida de ajedrez. También se almacenará la información relativa a los usuarios. Esto implica que todos los módulos deberán acceder a la base de datos para realizar ciertas operaciones, como se explicará con más detalle en los siguientes apartados.

4.2 Herramienta pobladora de base de datos

El diseño propio de la base de datos implica la necesidad de implementar una herramienta cuyo objetivo sea poblar la base de datos con información referente a las partidas a

incluir. Esta herramienta será un módulo independiente del proyecto, que se comunicará únicamente con el módulo de la base de datos. La conexión será directa, ya que se trata de una herramienta de escritorio que será utilizada en labores consideradas de gestión y administración de la base de datos, y cuyas funcionalidades se deberán mantener fuera del alcance de los usuarios de la aplicación principal de visualización de partidas.

La función principal de esta herramienta consistirá en permitir seleccionar un fichero de tipo PGN, extraer la información de él e introducirla en la base. Si el archivo PGN está conformado por más de una partida, la herramienta deberá extraer cada una de ellas del fichero y crear una entrada independiente en la base de datos para cada una de ellas. Se deberá permitir también configurar los parámetros de conexión a la base de datos, así como la ruta donde se almacenan los archivos PGN dentro del servidor.

4.3 Herramienta de visualización de partidas

Esta herramienta conforma la parte principal del proyecto y deberá realizar las siguientes funciones:

- Recuperación de partidas de la base de datos en función de una serie de parámetros de búsqueda.
- Selección de una partida y visualización de la misma.
- Introducción de anotaciones asociadas a fragmentos de la partida seleccionada en la base de datos.
- Recuperación de las anotaciones realizadas por el propio usuario y por otros sobre la partida seleccionada.

Este módulo será invocado directamente desde el módulo de gestión de usuarios, por lo que podrá identificar el usuario que ha accedido a la aplicación. También deberá conectar con la base de datos para recuperar listas de partidas, información de partidas específicas, anotaciones sobre partidas específicas e introducción de anotaciones por un usuario específico sobre una partida determinada. Por tanto, es necesario establecer el medio por el cual el este módulo se conectará a la base de datos. Las dos opciones que se barajan son una conexión directa a la base de datos, o una conexión mediante un servlet.

Hay un problema que podría presentarse en la aplicación y consiste en que muchos clientes tienen cortafuegos que impiden conexiones a determinados puertos que interpretan como desconocidos, como podría ser el puerto al que se realiza la conexión a la base de datos. Esto podría dificultar gravemente el correcto funcionamiento de la herramienta.

Por otra parte, también puede resultar peligroso, desde el punto de vista de seguridad, el conceder acceso a la base de datos desde cualquier máquina cliente que ejecute la aplicación.

El único inconveniente en la implementación de un servlet intermedio sería una ligera complicación en la implementación, pero sería bastante procedente a la vista de las ventajas que ofrecería.

Por consiguiente se toma como decisión de diseño la implementación de un servlet que conecte la herramienta con la base de datos, siempre que ésta requiera recuperar o introducir información en la base de datos.

4.4 Gestión de usuarios

En los requisitos iniciales se estipuló la necesidad de un sistema básico de gestión de usuarios. Las funciones principales que deben implementarse son registro, autenticación y modificación de datos de usuario.

Respecto a la función de autenticación, el módulo de gestión debe conectar con la base de datos durante el proceso, y si todo es correcto, comprobar que el usuario está registrado y la contraseña es correcta y redirigir directamente a la herramienta de visualización de partidas.

Cuando se realice el registro, este módulo deberá conectar con la base de datos, para comprobar que no haya ningún usuario registrado bajo el mismo nombre de usuario. Si todo es correcto se introducirán los datos del nuevo usuario en la base de datos y se deberá redirigir directamente a la herramienta de visualización de partidas.

También se permitirá la modificación de los datos introducidos durante el registro. Para ello se deberá conectar de nuevo con la base de datos para recuperar los datos de registro del usuario en cuestión. Serán presentados de forma que se puedan editar y si el proceso se realiza correctamente, se actualizará la base de datos con las modificaciones introducidas.

La forma de conexión para este módulo con la base de datos será de forma directa, ya que la relativa sencillez de las funciones realizadas permite implementar sistemas de conexión que oculten el contenido real de las operaciones ejecutadas sobre la base de datos a los usuarios.

Capítulo 5

Diseño de Alto Nivel

En este capítulo se analizarán aspectos globales del diseño de los distintos módulos que conforman la aplicación. Entre los aspectos a analizar se encuentran la elección del software a utilizar, el modelo de datos implementado, la interconexión de los distintos módulos, así como aspectos relevantes de diseño de los propios módulos.

5.1 Elección de software

En este apartado se expondrán las distintas opciones de software seleccionadas para cada función dentro de los distintos módulos de la aplicación. Se razonará la elección de una determinada herramienta entre las distintas posibilidades siempre que sea relevante.

Base de datos
Anteriormente, en el segundo capítulo, se ha analizado la oferta de sistemas de gestión de bases de datos existentes en el mercado. Se optará por la elección de un sistema de licencia libre. Entre los analizados, aquéllos que se distribuyen bajo algún tipo de licencia libre son: MySQL, PostgreSql y Firebird.

Las características de cualquiera de estos tres sistemas de gestión de bases de datos cumplirían los requisitos planteados inicialmente.

Debido a que la aplicación presentará una baja concurrencia en la modificación de datos y en cambio el entorno tenderá a ser intensivo en lectura de datos, MySQL se presenta como la opción más idónea para este tipo de aplicaciones. Además, otras de sus prestaciones mencionadas en la correspondiente entrada de wikipedia [9] que la convierten en la opción preferente son su soporte a gran cantidad de datos (puede llegar a almacenar hasta 50 millones de registros) y la implementación de mecanismos de conectividad segura.

Servidor web
La elección del servidor depende de varios factores, entre los que se incluyen el presupuesto, la tecnología de servidor que desea utilizar y las características proporcionadas por el servidor.

Respecto a la tecnología de servidor, el lenguaje de desarrollo elegido ha sido Java. En el libro “Java – A Beginner’s Guide” [21] se detallan una serie de características que hacen de Java la opción más adecuada para implementar la aplicación de este proyecto:

- Sencillo: Java es fácil de aprender, debido a su concisión y cohesión.
- Seguro: Java proporciona medios seguros de crear aplicaciones para Internet.

- Orientado a objetos: Java es eficiente en la reutilización de código, por lo que el desarrollo con Java resulta eficiente y rápido.
- Robusto: Java fomenta la programación libre de errores mediante sus tipos de datos y mediante comprobaciones en tiempo de ejecución.
- Multihilo: Java proporciona soporte integrado para la programación multihilo.
- Multiplataforma: Java no es dependiente de un tipo de máquina o de sistema operativo específicos.
- Eficiencia: Java presenta una elevada velocidad de ejecución.
- Distribuido: Java fue diseñado teniendo en cuenta el entorno distribuido de Internet.
- Dinámico: Los programas Java incluyen cantidades sustanciales de información en tiempo de ejecución que es usada para verificar y resolver el acceso a objetos durante la ejecución.
- Flexible: Java proporciona una gran cantidad de librerías y componentes, que permiten la implementación tanto de aplicaciones para Internet como aplicaciones de escritorio.

Entre los servidores de licencia libre Tomcat [22] se presenta como la opción más evidente y favorable, ya que es uno de los más utilizados en el mercado y está diseñado especialmente para el desarrollo web en Java. Tomcat (también llamado Apache Tomcat) funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat es un servidor web que implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Sun Microsystems [23]. Incluye el compilador Jasper, que compila páginas JSP convirtiéndolas en servlets. Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java.

Por tanto se utilizará el servidor Tomcat, en su versión 5.5 para el despliegue de los módulos de gestión de usuarios y la herramienta de visualización de partidas.

Herramienta de visualización de partidas

Anteriormente se han analizado las herramientas existentes y las tecnologías en que están implementadas: JavaScript y *applet* Java.

Las aplicaciones analizadas basadas en JavaScript tienen un funcionamiento y una estructura muy similar. A partir de una aplicación de escritorio se genera un archivo HTML y un archivo de script. Éste último nunca cambia. Sólo se modifica el archivo HTML, que contiene los movimientos de la partida vinculados a llamadas al código javascript, las cuales controlan el comportamiento del tablero.

```
<span class="V1">[<a href="javascript:c(0,11)" ID="l11">4...Qf5+</a> <a href="javascript:c(0,12)" ID="l12">5.Kg1</a>
 <a href="javascript:c(0,13)" ID="l13">Qb1+</a> <a href="javascript:c(0,14)" ID="l14">6.Kg2</a> <a href="javascript:c(0,15)" ID="l15">Qe4+</a>
 <a href="javascript:c(0,16)" ID="l16">7.Kh3</a> <a href="javascript:c(0,17)" ID="l17">Qf5+</a> <a href="javascript:c(0,18)" ID="l18">8.g4</a>
 <a href="javascript:c(0,19)" ID="l19">Qf1+</a> <a href="javascript:c(0,20)" ID="l20">9.Kg3</a> <a href="javascript:c(0,21)" ID="l21">Qe1+</a>
 <a href="javascript:c(0,22)" ID="l22">10.Kf3</a> <a href="javascript:c(0,23)" ID="l23">Qf1+</a> <a href="javascript:c(0,24)" ID="l24">11.Ke3</a>
 <a href="javascript:c(0,25)" ID="l25">Qc1+</a> <a href="javascript:c(0,26)" ID="l26">12.Kf2</a>]</span></td></tr>
<tr><td COLSPAN=2>
<span class="V0">
<a href="javascript:c(0,28)" ID="l28">5.Qe7</a>
</span>
```

Figura 5.1.1 Ejemplo de extracto de código JavaScript generado por la aplicación pgntojs

Por lo tanto, para reutilizar el código JavaScript de estas aplicaciones sería necesario generar un archivo HTML por cada fichero PGN que se fuera a mostrar. Esto significa que con JavaScript se requeriría la implementación de esta funcionalidad en la herramienta, aunque no sería una tarea excesivamente compleja.

Actualmente la mayoría de los navegadores han solucionado los inconvenientes que surgían con cada una de las dos tecnologías, y el comportamiento por consiguiente es muy eficiente. De hecho, el tiempo de carga para ambas tecnologías es pequeño, unos dos segundos, y muy similar para las dos.

Uno de los inconvenientes que podría tener JavaScript es que se envía al cliente el código fuente, pero esto no nos supondría problema alguno, al ser aplicaciones de código libre.
Por otra parte, los applets sí nos podrían ofrecer más potencia a la hora de modificar la interfaz gráfica, ya que disponemos de la potencia del lenguaje Java.

Por lo tanto, para este módulo, se considera como mejor elección utilizar como tecnología un *applet* Java, como ya se apuntó anteriormente, y entre las herramientas estudiadas basadas en *applet* Java se elegiría Sjkbase como la más apropiada.

Gestión de usuarios
Hay numerosos lenguajes de programación y tecnologías que permiten la implementación de serie de páginas que cumplan los requisitos planteados inicialmente y que oculten el contenido real de las transacciones de la base de datos.
Al estar trabajando con herramientas basadas en Java, lo más apropiado es utilizar tecnología Java para generar las páginas web que permitan las operaciones anteriormente descritas para el módulo de gestión de usuarios. Por tanto se utilizarán páginas JSP para tal efecto, ya que son un medio muy extendido, eficaz y sencillo de generar contenido web dinámico.

5.2 Modelo de datos

Nuestro modelo de datos consta de tres entidades principales: las partidas, que ya hemos decidido que serán en formato PGN; las anotaciones que queremos poder permitir sobre las partidas, y los usuarios, que interactuarán con la herramienta y podrán consultar partidas y realizar las anotaciones. Respecto a las anotaciones, queremos que cada usuario pueda añadirlas sobre diversos fragmentos de la partida: rangos de movimientos

(pudiendo distinguir entre movimientos de blancas y de negras) y posiciones, y sobre partidas enteras. Por tanto, se podría representar el diagrama relacional de nuestra base de datos como se muestra en la siguiente figura.

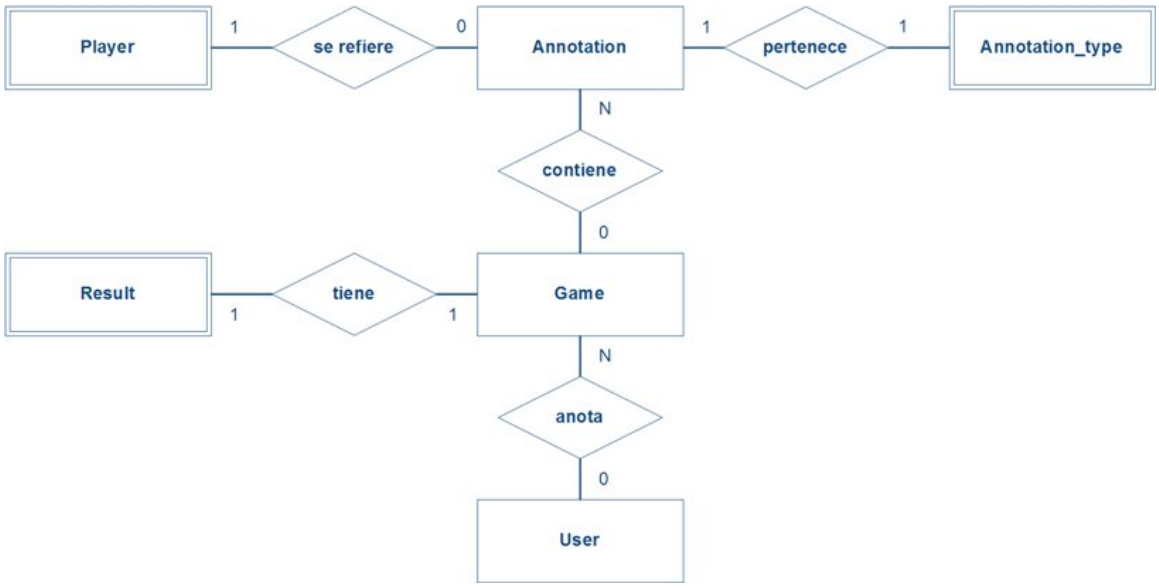
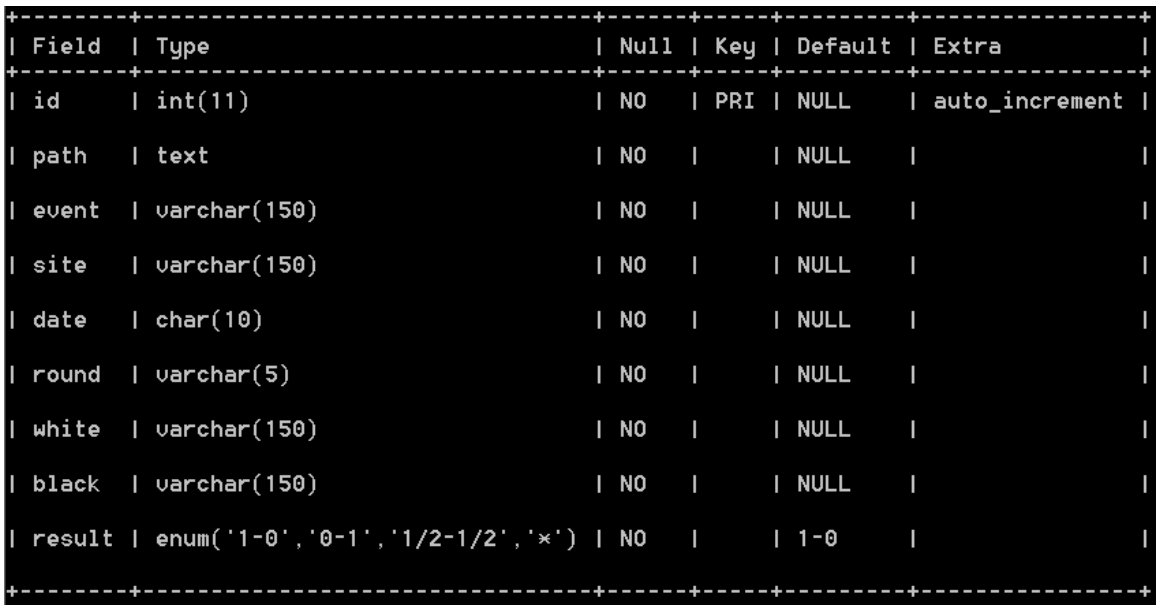


Figura 5.2.7 Diagrama relacional del diseño de la base de datos.

- Tabla game: es la tabla en la que se almacena la información relativa a las partidas. Se decidió no almacenar en esta tabla los ficheros enteros, sino únicamente la ruta relativa a los ficheros (campo *path*), los cuales se almacenan directamente en el servidor. Cada partida debe tener un identificador único (campo *id*). También se almacena en la tabla la información correspondiente a las siete etiquetas obligatorias del formato PGN en los siguientes campos:
  - *event*: el nombre del torneo.
  - *site*: el lugar donde el evento se llevó a cabo.
  - *date*: la fecha de inicio de la partida.
  - *round*: La ronda original de la partida.
  - *white*: El jugador de las piezas blancas.
  - *black*: El jugador de las negras.
  - *result*: El resultado del juego. Sólo puede tener cuatro posibles valores: "1-0" (las blancas ganaron), "0-1" (Las negras ganaron), "1/2-1/2" (Tablas), o "*" (para otro, ejemplos: el juego está actualmente en disputa o un jugador falleció durante la partida).



Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
path	text	NO		NULL	
event	varchar(150)	NO		NULL	
site	varchar(150)	NO		NULL	
date	char(10)	NO		NULL	
round	varchar(5)	NO		NULL	
white	varchar(150)	NO		NULL	
black	varchar(150)	NO		NULL	
result	enum('1-0','0-1','1/2-1/2','*')	NO		1-0	

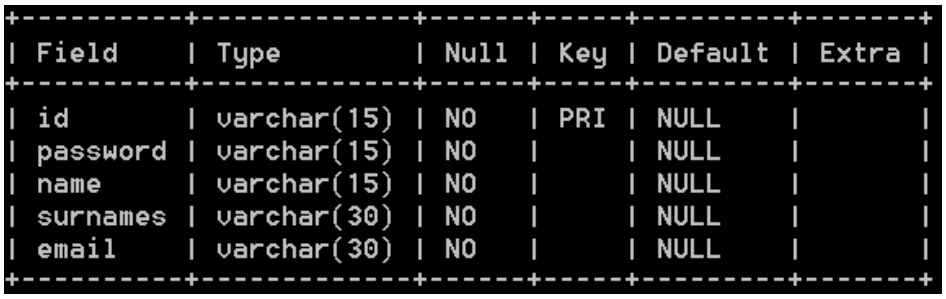
Figura 5.2.1 Resultado del comando "describe game" en mysql.

- Tabla annotation: es la tabla donde se almacena la información relativa a cada anotación. Cada anotación debe tener un identificador único (campo *id*). También se almacena el texto de la anotación, el identificador del usuario que realiza la anotación, el de la partida sobre la que se realiza, el tipo de comentario y el rango (campos *begin* y *end*).

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
value	text	NO		NULL	
user_id	varchar(15)	NO	MUL	NULL	
game_id	int(11)	NO	MUL	NULL	
annotation_type	enum('G','M','P')	NO		G	
begin	int(11)	YES		NULL	
end	int(11)	YES		NULL	
player	enum('W','P','W+B')	NO		W+B	

Figura 5.2.2 Resultado del comando "describe annotation" en mysql.

- Tabla user: es la tabla donde se almacena la información relativa a cada usuario. Utilizamos el nombre de autenticación como identificador único. Almacenamos también la contraseña elegida para acceder a la aplicación. Y añadimos campos relativos a la información personal de cada usuario, como nombre, apellidos y dirección de correo electrónico.



Field	Type	Null	Key	Default	Extra
id	varchar(15)	NO	PRI	NULL	
password	varchar(15)	NO		NULL	
name	varchar(15)	NO		NULL	
surnames	varchar(30)	NO		NULL	
email	varchar(30)	NO		NULL	

Figura 5.2.3 Resultado del comando "describe user" en mysql.

5.3 Herramienta de visualización de partidas

En este apartado se pasará a considerar el diseño de la herramienta de visualización de partidas. Una vez determinada la tecnología que se utilizará, se procederá a seleccionar la herramienta que se usará como base de la aplicación y las modificaciones gráficas necesarias para cumplir con las funcionalidades planteadas en los requisitos iniciales.

Elección de applet

Ya se ha planteado la idoneidad de la utilización de un *applet* Java frente a código de Javascript. Entre los applets analizados, la mayoría leen directamente los ficheros PGN y despliegan la partida en el visor, por lo que se ahorraría la implementación de la funcionalidad que genera el HTML. Y de todos ellos, al tener características muy similares, las opciones más razonables serían MyChessViewer y Sjkbase ya que el autor proporciona el código fuente bajo licencia libre, ambos desarrollados en AWT. De entre ellos el segundo posee un código mejor estructurado, está dividido en 15 clases Java con funcionalidades bien delimitadas, mientras que el primero está compuesto únicamente por 3, con una clase que alcanza las tres mil líneas de código. Además, en el navegador en el que fueron probados, el comportamiento de Sjkbase fue superior, ya que MyChessViewer producía un cierto parpadeo al realizar algunas operaciones. Por lo tanto, entre los applets, la elección óptima se evidenciaría ser la herramienta Sjkbase.

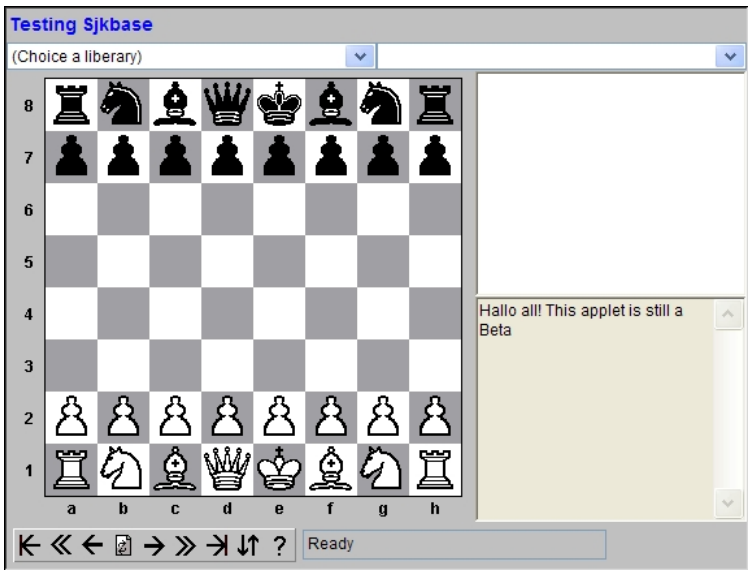


Figura 5.3.1 Fuente: http://home.online.no/~malin/sjakk/download/sjkbase11/sjkbase.html

Una vez elegida la herramienta sobre la que basar la aplicación, se deberá determinar los cambios que deberán realizarse sobre la misma, a fin de cumplir con los requisitos planteados inicialmente.

Como puede apreciarse en la imagen anterior, la herramienta sjkbase permite elegir entre una serie de librerías y, una vez cargada una librería, elegir entre un conjunto de partidas. Las librerías consisten en ficheros PGN que contienen varias partidas. Al seleccionar uno de ellos, el programa lee el fichero y muestra en el combo de la derecha todas las partidas contenidas en el fichero, permitiendo seleccionar una de ellas. Los movimientos contenidos en la partida son mostrados en el cuadro superior derecha, mientras que los comentarios contenidos en el archivo PGN son mostrados en el cuadro inferior derecha. Se puede navegar por la partida, bien pinchando directamente sobre el movimiento al que se quiera desplazar la partida, bien mediante la barra con botones que aparece debajo del tablero.

Varias de las características de dicha herramienta incumplen alguno de los requisitos iniciales. La primera es la inclusión de varias partidas dentro de un archivo PGN, ya que uno de los requisitos era que cada archivo contendría una única partida.

Por otro lado, tampoco se permite la búsqueda por parámetros de partida, otro de los requisitos previamente enumerados.

Y por último, como es evidente, no aparece ningún mecanismo que permita introducir y recuperar anotaciones. También es bastante mejorable la interfaz gráfica, pudiendo hacerse más atrayente para el usuario.

A continuación se analizarán los principales cambios introducidos en la herramienta.

Cambios en la búsqueda de partidas
Como ya se ha mencionado, era necesario sustituir el mecanismo de selección de partidas de la herramienta original por un motor de búsqueda basado en parámetros relativos a las partidas. Por tanto se vio necesario sustituir las dos cajas de selección (librería y partida) por una serie de campos que permitieran realizar una búsqueda a partir de las propiedades obligatorias de una partida en formato PGN.

También se consideró interesante la posibilidad de habilitar la búsqueda en función de las anotaciones introducidas por los usuarios, por lo que se han habilitado campos para tal propósito.

Una vez introducidos los parámetros se debería pinchar sobre el icono que representa una lupa y si se encontrara alguna coincidencia con los parámetros introducidos, se mostraría en el cuadro superior derecho. Este cuadro despliega la lista de partidas en grupos de cinco. Si hubiera más de cinco partidas que coincidan con los parámetros, la barra de botones encima del cuadro permite navegar por la lista completa de partidas recuperadas. Si se selecciona una de las partidas, haciendo clic, se cargará la partida, mostrándose la lista de movimientos en el cuadro inferior. En el cuadro de en medio se muestra la información contenida en las siete etiquetas básicas de la partida.

Una vez cargada la partida, el sistema de navegación es el mismo del applet original: pinchando directamente sobre el movimiento deseado o mediante la barra de botones de la parte inferior.

Implementación de anotaciones
Las anotaciones sobre fragmentos de una partida son la funcionalidad sobre la que se basa este proyecto, por lo que debe ocupar una parte importante en el diseño gráfico de la aplicación.

Se han añadido dos áreas de texto a la izquierda de la aplicación, además de una serie de botones de radio y dos combos. Hay un botón de radio por cada uno de los tres tipos de anotación: partida, posición y movimiento. Si se selecciona el botón posición aparecen otros dos botones de radio que permiten seleccionar entre blancas o negras. Los dos combos permiten seleccionar los puntos de inicio y fin de la anotación, cumpliendo de esta forma el requisito de la anotación sobre rangos de fragmentos. Estos combos no están activados cuando el botón de partida sí lo está, ya que los rangos sólo aplican a posiciones y movimientos. El usuario puede introducir el texto de la anotación en el campo de texto de la parte superior izquierda. Para que la anotación sea almacenada en la base de datos, el usuario debe presionar el icono del bocadillo.

En el área de texto de la parte inferior izquierda se muestran todas las anotaciones introducidas por los usuarios para la partida seleccionada. Se pueden observar a su vez unos botones de navegación. Estos botones permiten ir cambiando entre todos los usuarios que han introducido cualquier tipo de anotación para la partida seleccionada, incluido el usuario actual.

Otros cambios
El resto de modificaciones que se ha visto necesario implementar son relativas al atractivo gráfico de la aplicación.

El primero de ellos es un cambio en el color de fondo, que ha pasado de un gris oscuro a un azul claro. Este cambio resulta bastante inmediato, ya que el *applet* Java original permitía seleccionar el color de fondo de la aplicación como uno de los parámetros del applet. Otros parámetros que incluía eran el idioma y la imagen de la que se toman las piezas de ajedrez para el tablero. El que estas opciones estuviesen implementadas planteaba la posibilidad de utilizarlas para aportar un mayor poder de configuración de la aplicación al usuario.

Por tanto se habilitó un menú en la parte superior de la aplicación donde se ofrece la opción de modificar tanto el tipo de piezas como el idioma de la aplicación.

Para el tipo de piezas, se incluye un fichero en el servidor listando el nombre de todas las imágenes que contienen las distintas opciones de piezas. El applet lee dicha lista y muestra una entrada en el submenú para cada una de ellas cuando se pulsa sobre la opción “Piezas”. El applet carga la nueva imagen y recarga el tablero para mostrarlo con las nuevas piezas seleccionadas.

En cuanto a los idiomas, el applet incluía un fichero de configuración para cada uno de los siguientes idiomas: inglés (por defecto), francés, español, neerlandés y alemán. Como ya se ha mencionado, el idioma era configurable a través de los parámetros del applet. Por tanto se ha introducido una opción en el menú “Idiomas” para cada uno de ellos. Cuando se pulsa se recarga el contenido del applet y se muestra en el nuevo idioma. Es necesario comentar que ha sido necesario incluir una nueva entrada en los ficheros de configuración de idioma por cada una de las etiquetas nuevas introducidas en la aplicación, tanto en el tablero de búsqueda, como en la parte de las anotaciones y en el propio menú.

El último de los cambios es el de los botones de navegación de la partida, botones que también se han utilizado para la navegación por la lista de partidas recuperadas en la búsqueda y por las anotaciones introducidas por los usuarios. Se ha procedido a sustituir las imágenes para los botones que incluía la aplicación por un conjunto de imágenes para dichos botones más modernas y atrayentes.

Por tanto, después de los cambios realizados la apariencia del applet es la que se muestra en la siguiente figura, en la que puede observarse una notable diferencia con respecto a la de la figuras 5.3.1.

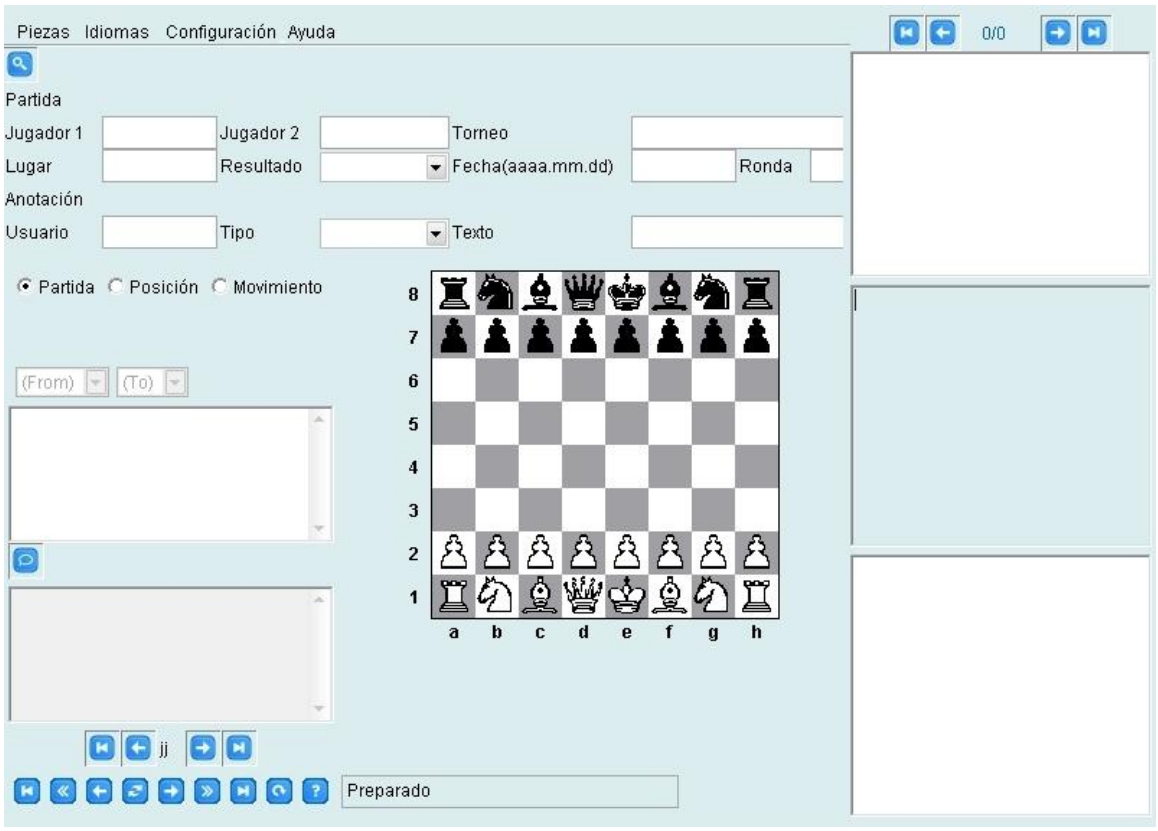


Figura 5.3.2 Captura de la herramienta una vez modificada

Modificaciones posteriores

El conjunto de modificaciones anteriormente descrito fue realizado con el propósito de adaptarse a los requisitos funcionales para la aplicación. Pero una vez implementados, se estimó conveniente realizar otra serie de cambios con la finalidad de que la aplicación fuese más intuitiva y manejable para el usuario, como se muestran en la figura 5.3.3.

Uno de los cambios es relativo al panel de búsqueda. Debido a que el panel implementado ocupaba gran parte del espacio de la aplicación, se ha optado por implementar un panel que incluya únicamente tres campos, los referentes a los jugadores y al torneo. Pero no se ha eliminado la búsqueda por el resto de parámetros, sino que se ha añadido un botón, de búsqueda avanzada, que muestra el panel con todos los campos que había anteriormente, como se mostraba en la figura 5.3.2, con la salvedad de que el botón de búsqueda avanzada se sustituye por uno de búsqueda simple. Al pulsarse este botón de búsqueda simple, se vuelve a la versión simplificada del panel.

También se ha modificado la parte de las anotaciones. Debido a que su funcionamiento podría resultar poco intuitivo para el usuario, se ha optado por eliminar el área de texto donde se introducían las anotaciones, así como los botones de radio y los combos de la ventana principal del applet, y en su lugar añadir un botón que al presionarlo lanza una ventana conteniendo todos los campos, como puede apreciarse en la imagen 5.3.3. Al

pulsar el botón “OK” en esta nueva ventana se introduciría la anotación, mientras que al pulsar el botón “Cancel” se cerraría la ventana sin introducir anotación alguna.

El último cambio tiene que ver con la lista de movimientos de la partida. Se ha reubicado en el espacio que ocupaba anteriormente el panel de los comentarios, compensándose de ese modo la cantidad de elementos en la ventana a la derecha y a la izquierda. Por otro lado, se ha implementado un mecanismo mediante el cual, cuando un movimiento o una posición contienen un comentario por parte de alguno de los usuarios, ese movimiento o posición aparecen en color verde, facilitándose así al usuario el acceso a los fragmentos de la partida con anotaciones. Evidentemente, para las anotaciones referentes al total de la partida no es necesario resaltar nada en verde, ya que aparecen en el panel de anotaciones siempre que existen.

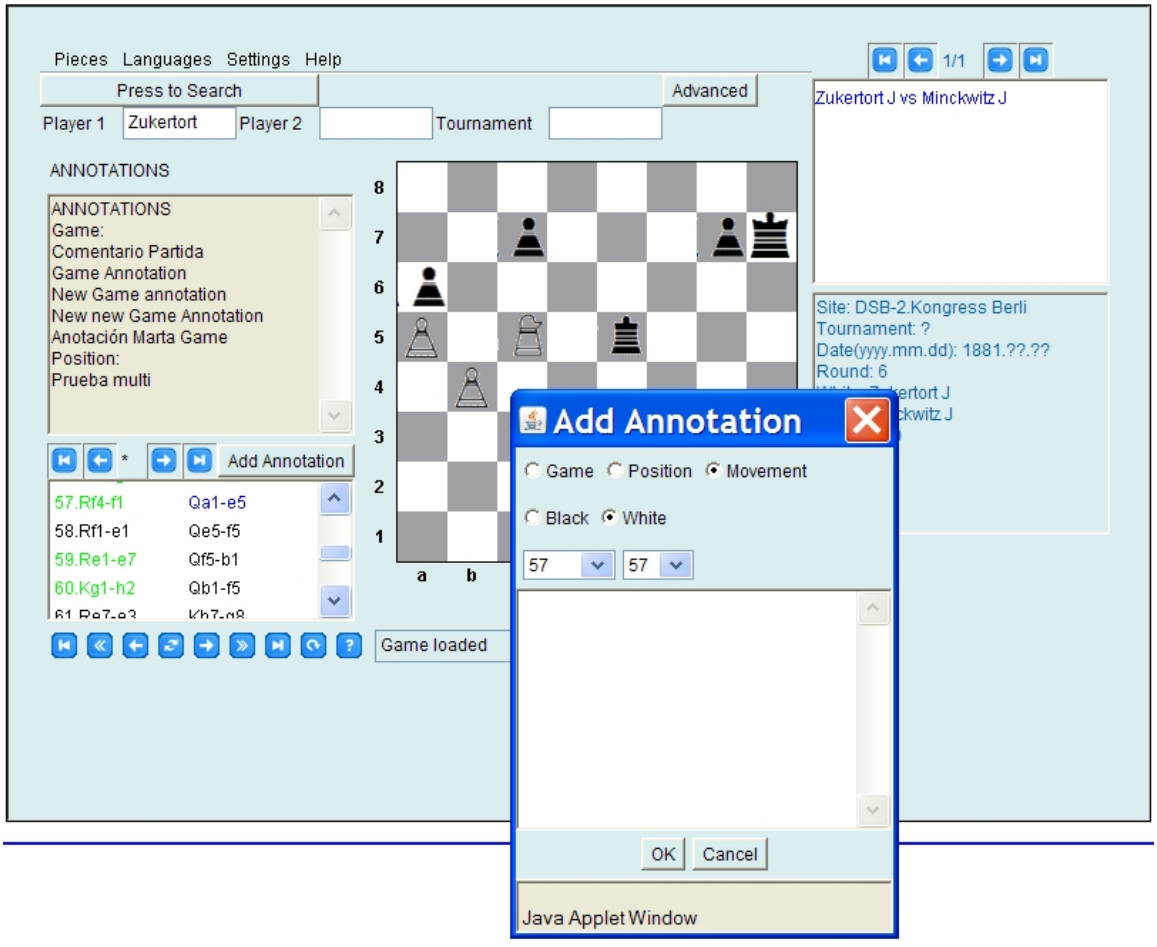


Figura 5.3.3 Captura de la herramienta después del segundo conjunto de modificaciones

5.4 Control de acceso

El sistema de gestión de usuarios restringe el acceso a la aplicación sólo a los usuarios que hayan procedido al registro en la misma con anterioridad. El acceso se realiza

mediante la función de autenticación. El módulo de gestión implementa una serie de JSP's que conectan con la base de datos para obtener de ella la información necesaria para el control de acceso. En la autenticación se comprueba que el usuario existe y que la contraseña es correcta, produciéndose entonces el acceso a la herramienta de visualización de partidas.

Otra forma de acceder es completando correctamente el proceso de registro. Durante este proceso se debe comprobar que no haya ningún usuario registrado bajo el mismo nombre de usuario. Para ello se realiza una conexión a la base de datos. Si existiera ya un usuario con el mismo nombre se indica de forma que se pueda introducir un nombre distinto.

Tanto las páginas JSP del módulo de gestión de usuarios como el applet de la herramienta de visualización de partidas se conectan a la base de datos para recuperar e introducir información. La diferencia entre ellas es que el applet realiza la conexión mediante un servlet intermedio y las JSP's no. Esto se debe a que el código del applet se envía al navegador que lo ejecuta compilado, de manera que un usuario malicioso podría tener acceso al código que realiza las operaciones en la base de datos si éste se invocara directamente en el applet. Al implementar un servlet intermedio se procura una capa extra que proporciona una mayor medida de seguridad. Por el contrario el código de las JSP's es compilado en el servidor, donde se crea un servlet por cada página. Por tanto la implementación del código de la base de datos queda oculta a los ojos de los usuarios.

La conexión a base de datos está configurada mediante los respectivos ficheros de configuración dentro del servidor Tomcat, analizados con más detalle en el Apéndice B.

5.5 Comunicación entre el applet y el servlet

Como ya se ha apuntado anteriormente, el applet se conecta a la base de datos mediante un servlet, de modo que la seguridad de la comunicación es mayor que si se conectara directamente. El applet necesita conectarse con la base de datos para las siguientes operaciones, las cuales distingue en la conexión al servlet mediante un comando distinto para cada una de ellas:

- Recuperar una lista de partidas: cuando el usuario efectúa una búsqueda, introduce una serie de parámetros sobre la que se basará esta búsqueda. El applet conecta con el servlet indicándole el tipo de operación y proporcionándole todos los parámetros que conforman la búsqueda. El servlet conecta con la base de datos y recupera las partidas pertinentes, devolviéndolas de vuelta al applet.

- Recuperar una lista de anotaciones: cuando el usuario selecciona una partida, el applet debe recuperar todas las anotaciones relativas a la misma. Para ello el applet conecta con el servlet indicándole el tipo de operación y enviándole el identificador de la partida seleccionada. El servlet conecta con la base de datos y recupera las todas anotaciones asociadas a dicha partida, devolviéndolas de vuelta al applet.

- Añadir una anotación: cuando el usuario añade una anotación, el applet conecta con el servlet para indicarle el tipo de operación que debe realizar, y el valor de los campos de la anotación que se introducirán en la base de datos tras la llamada a ésta del servlet. En este caso el servlet no devuelve nada al applet.

La recuperación de un fichero PGN específico no se realiza mediante ningún comando del servlet. Ya que es un fichero real con una ruta propia dentro del servidor, se abre una conexión HTTP con la URL del fichero, la cual se obtuvo con la lista resultado de la búsqueda, y se descarga el fichero, el cuál es leído y desplegado como una partida en el tablero por la herramienta.

Capítulo 6

Diseño de Bajo Nivel

En este apartado se detallarán particularidades de implementación de los distintos módulos que conforman la aplicación. Se describirán sólo los métodos y clases principales, que aporten información relevante sobre la implementación de las funcionalidades básicas en cada módulo de la aplicación, añadiéndose porciones limitadas de código si se encontrase conveniente. En el caso de fragmentos más extensos de código serán incluidos en el apéndice y se hará referencia a ellos en este apartado. También se describirán operaciones esenciales en el funcionamiento de la aplicación, tales como la comunicación con la base de datos a través del servlet, o la lectura de los ficheros PGN. Siempre que sea posible se acompañarán diagramas de clases que reflejen las interrelaciones de las distintas unidades de código.

6.1 Estructura del applet

Como ya se ha explicado anteriormente, se ha procedido a reutilizar una herramienta ya existente, basada en un *applet* Java. La estructura de tal herramienta no ha sido modificada, sino que únicamente se han introducido una serie de cambios en el código e implementado algunas clases auxiliares. En este apartado se aportarán detalles de implementación que rigen el funcionamiento del applet y la interacción con el usuario. Los detalles de implementación de los objetos gráficos específicos para añadir las funcionalidades estipuladas en los requisitos previos no se analizarán en este apartado, sino en el siguiente.

En la siguiente figura podemos observar la representación del applet en diagrama UML y apreciar las relaciones existentes entre cada una de sus clases principales.

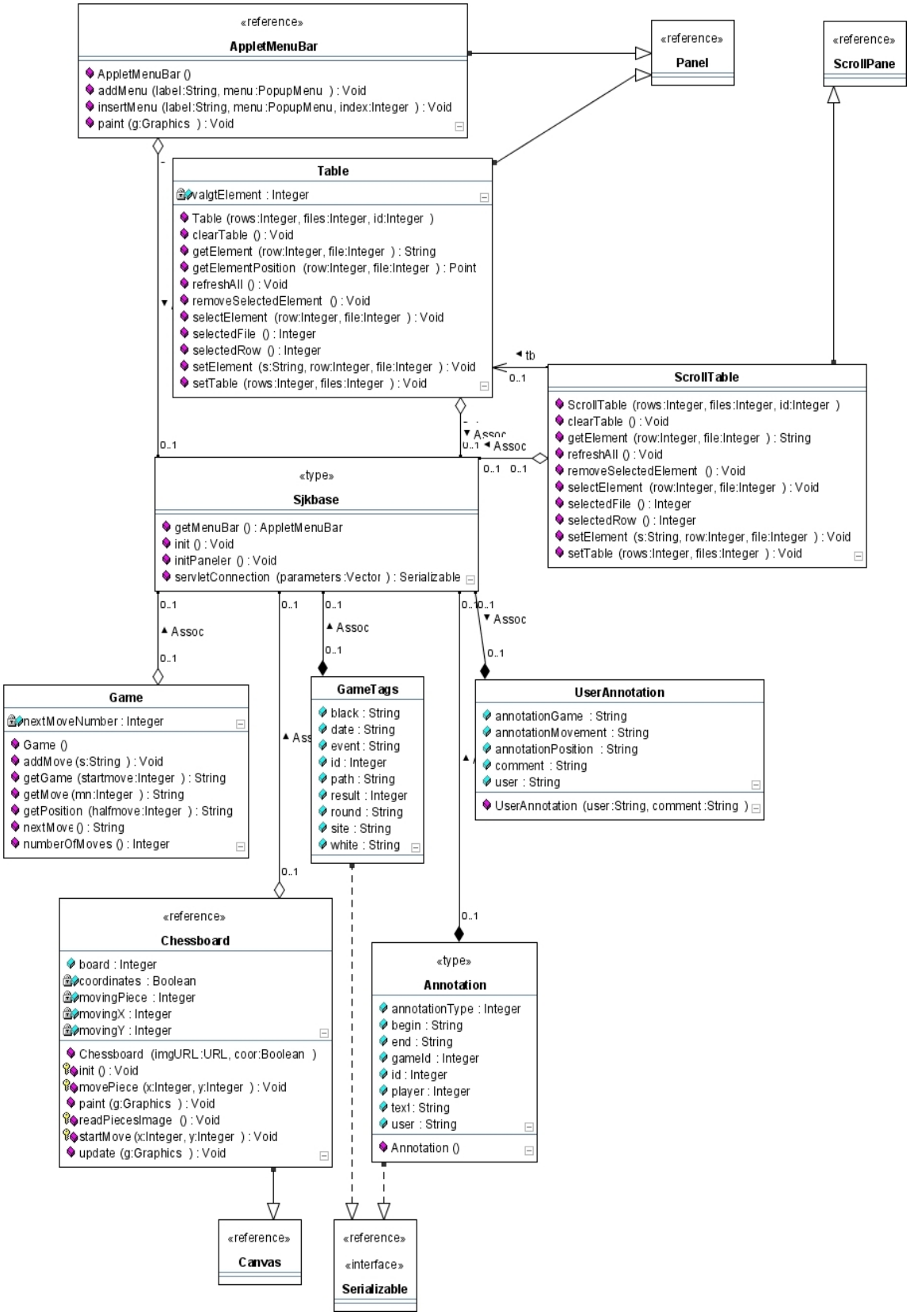


Figura 6.1.1 Diagrama UML del applet

El diseño de la herramienta sjkbase se basa en una clase principal, llamada también Sjkbase, que extiende de la clase Applet y que concentra la mayor parte del código del programa. Esta clase lee los parámetros que se pasan al applet, configurando mediante ellos el idioma, el color de fondo y las imágenes para las piezas. Estos parámetros son los siguientes:

- Pieces: nombre del archivo de imagen que contiene las piezas de ajedrez. A partir del valor se crea un objeto URL que almacena la ruta a dicho archivo.

- Bgcolor: código de color que se fijará como fondo de la aplicación. Se crea un objeto Color, donde se almacena, a partir de este valor.

- Coordinate: si el valor de este parámetro es “yes” o “no”, se modifica el valor de la variable coordinate, que indicará si se muestran coordenadas en el tablero o no.

- Quiz: si el valor de este parámetro es “yes” o “no”, se modifica el valor de la variable quiz, que indicará si se muestra el valor de los movimientos subsiguientes en la partida o no.

- Language y Country: a partir del valor de estos parámetros se crea un objeto de la clase LocaleText, que carga el fichero de configuración de lenguaje correspondiente.

La clase Sjkbase contiene un método que se inicializa todos los componentes gráficos del applet. Al applet se le añaden entonces una serie de paneles y objetos que contienen toda la funcionalidad de la aplicación:

- Barra de menú: objeto AppletMenuBar que se dibuja en la parte superior del applet. Se detallará su funcionamiento en el siguiente apartado.

- Panel de atributos: se dibuja en la parte derecha el panel Kommentarfelt, que muestra los principales atributos de la partida cuando se carga.

- Panel de búsqueda: panel que se añade en la parte superior izquierda del applet, debajo de la barra de menú. Contiene los parámetros de búsqueda de la partida y un área donde se muestran los resultados de la búsqueda. Se ampliarán detalles de su funcionamiento en el siguiente apartado.

- Panel de anotaciones: panel que implementa la funcionalidad que permite introducir anotaciones. Se detallará su funcionamiento en el siguiente apartado.

- Panel de posiciones: panel que contiene el objeto ScrollTable, objeto que extiende de la clase ScrollPane y que contiene un objeto Table, un Panel que se divide en filas y columnas donde se añaden elementos, que en este caso corresponden con la lista de movimientos de la partida. Un objeto de la clase interna MovePanelListener, que implementa la interfaz ActionListener, se añade como listener al objeto de la clase ScrollTable. Cuando se lanza un evento, lo que significa que el usuario ha pulsado

- Tablero de ajedrez: panel que contiene una instancia de la clase Chessboard, clase que implementa el tablero de ajedrez. Esta clase extiende de la clase Canvas y sobrescribe su método paint. Este objeto lee la imagen con las piezas para poder desplegarlas, dibuja el tablero y guarda un array de enteros con la posición de cada pieza en el tablero. Cada vez que en el applet se lanza un evento que modifica la disposición del tablero, se invoca un método de la clase Chessboard que modifica la posición de las piezas del tablero a partir del código FEN pasado como parámetro y llama al método repaint de la clase Canvas, que repinta el tablero de ajedrez.

- Barra de botones del tablero: es la barra de botones situada debajo del Panel que contiene las posiciones de la partida. La clase ButtonListener, que implementa la interfaz ActionListener, controla su comportamiento. Cuando se lanza un evento, lo que significa que el usuario ha pulsado sobre uno de los botones, se identifica, a partir del evento lanzado, el botón que lo ha provocado y se realiza una acción diferente para cada botón. Se retrocede al inicio o se avanza al final, y se retroceden o avanzan uno o cinco movimientos respectivamente. También se puede refrescar la posición actual o girar el tablero. El código calcula la posición a la que se debe cambiar, y llama al método de la clase Chessboard que modifica la posición.

- Barra inferior: panel que contiene el campo de texto no editable donde se van mostrando los mensajes asociados a las acciones realizadas por el usuario.

6.2 Diseño de la interfaz gráfica del applet

Los elementos del applet original que permanecen en el nuevo diseño son, además del tablero, el Panel donde se muestran los atributos de una partida y el Panel seleccionable donde se muestran los movimientos de la partida, aunque con ciertas modificaciones. En el primero ya no se muestran los comentarios internos de la partida, sino que se muestran junto a las anotaciones. Y en el segundo se ha implementado un mecanismo mediante el cual, cuando un movimiento o una posición contienen un comentario por parte de alguno de los usuarios, ese movimiento o posición aparecen en color verde, facilitándose así al usuario el acceso a los fragmentos de la partida con anotaciones. Evidentemente, para las anotaciones referentes al total de la partida no es necesario resaltar nada en verde, ya que aparecen en el panel de anotaciones siempre que existen.

Para el resto de funcionalidades ha sido necesario hacer un diseño gráfico que ya se ha esbozado en el capítulo anterior y cuyos detalles de implementación pasarán a tratarse a continuación.

Panel de búsqueda

Implementación de un panel en la parte superior de la aplicación para la búsqueda de partidas. Este panel cuenta con campos de texto donde introducir los parámetros, un botón para lanzar la búsqueda y un área donde se muestran los resultados de la búsqueda. Como esta área debe ser seleccionable, ya que se debe poder cargar la partida elegida, se ha optado por utilizar el objeto Table implementado por la herramienta original. Al seleccionarse uno de las partidas de esta lista, el applet invoca a un método de la clase PGNFile, que lee un fichero en formato PGN a partir de su URL y crea un objeto Game, el cual representa una partida y guarda toda la información referente a ella. El appet utiliza este objeto Game, para obtener el código de la posición de cada movimiento, y actualizar así el objeto gráfico ChessBoard. Los parámetros de búsqueda son los siguientes:

- Torneo: el nombre del torneo o de la competición.
- Lugar: el lugar donde el evento se llevó a cabo
- Fecha: la fecha de inicio de la partida en formato AAAA.MM.DD.
- Ronda: La ronda original de la partida.
- Blancas: El jugador de las piezas blancas.
- Negras: El jugador de las negras.
- Resultado: El resultado del juego. Sólo puede tener cuatro posibles valores: "1-0" (las blancas ganaron), "0-1" (Las negras ganaron), "1/2-1/2" (Tablas), o "*" (para otro, ejemplos: el juego está actualmente en disputa o un jugador falleció durante la partida).

Para los campos “Torneo”, “Lugar”, “Fecha”, “Ronda”, “Blancas” y “Negras”, se proporciona un campo de texto que permite al usuario introducir libremente el texto de búsqueda. En la etiqueta que acompaña al campo “Fecha” se indica el formato que debe tener la fecha. Para los campos “Blancas” y “Negras” se ha fijado el texto como “jugador 1” y “jugador 2”, con la intención de que se recuperen las partidas para las que el nombre de sus jugadores coincida con cualquiera de estos dos parámetros de búsqueda, independientemente de cuál de los dos jugara con blancas y cuál con negras. Para el campo “Resultado”, por el contrario, se habilita una caja de selección o combo, que permite elegir entre los únicos cuatro valores permitidos.

Para tal efecto se incluyen tres campos adicionales:

- Usuario: usuario que ha introducido la etiqueta.
- Tipo: El tipo de anotación. Sólo puede tener valores: partida, posición y movimiento.
- Texto: contenido de la anotación realizada por el usuario.

Al igual que antes, para los campos “Usuario” y “Texto”, se proporciona un campo de texto que permite al usuario introducir libremente el texto de búsqueda. Para el campo “Tipo”, por el contrario, se habilita una caja de selección, que permite elegir entre los únicos tres valores permitidos.

Al iniciarse el applet se muestran únicamente tres campos, los referentes a los jugadores y al torneo más un botón, de búsqueda avanzada. Al pulsar este botón se recarga el Panel que contiene los parámetros de búsqueda, añadiéndosele la totalidad de los campos, con la salvedad de que el botón de búsqueda avanzada se sustituye por uno de búsqueda simple y repintándose el applet. Al pulsarse este botón de búsqueda simple, se vuelve a la versión simplificada del panel, para lo que se vuelve a recargar el Panel y repintar, pero con los campos iniciales.

Al pulsarse el botón "Pulsa para Buscar", el applet invoca al servlet, el cual realiza una consulta a la base de datos a partir de los parámetros introducidos. Se devuelve una lista de GameTags, objeto Java que almacena los campos obligatorios de una partida en formato PGN, así como la id asociada en la base de datos y el nombre real del archivo donde está almacenada la partida. A partir de esta lista se despliegan en el área seleccionable, objeto Table, las partidas recuperadas de cinco en cinco. Con los botones centrales de la barra en la parte superior de la lista se van cargando las cinco partidas anteriores o las cinco siguientes, o bien se avanza al final de la lista o al principio, si se pulsan los botones de los extremos.

Cuando se selecciona una partida de la lista de partidas recuperadas, el applet accede al fichero por conexión http y lo transforma en un objeto Game. En ese momento vuelve a conectar con el servlet para recuperar las anotaciones relativas a dicha partida como una lista de objetos Annotation, los cuales representan una anotación almacenada en la base de datos con toda la información disponible en la base. A partir de esa lista de anotaciones se genera otra lista con objetos UserAnnotation, los cuales representan el conjunto de anotaciones relativas a un usuario para una posición dada de la partida y que también incluyen los comentarios que pudiera haber en el propio fichero PGN, para cada posición. Esta lista es con la que trabaja el applet para mostrar las anotaciones en el Panel correspondiente.

Barra de menú

Está implementada por un objeto AppletMenuBar. Se dibuja en la parte superior del applet y contiene las siguientes opciones de menú:

- Piezas: la aplicación ofrece la opción de modificar el tipo de piezas que se dibuja en el tablero de ajedrez. Al pulsar en el menú "Piezas" aparece una lista con los tipos de figuras disponibles. Para el tipo de piezas, se incluye un fichero en el servidor listando el nombre de todas las imágenes que contienen las distintas opciones de piezas. El applet lee dicha lista y muestra una entrada en el submenú para cada una de ellas. El applet carga la nueva imagen y recarga el tablero para mostrarlo con las nuevas piezas seleccionadas.

- Idiomas: Para cada uno de los idiomas disponibles aparece una entrada en el menú "Idiomas". Los idiomas disponibles son los que incluía el applet original en un fichero de configuración para cada uno de los siguientes idiomas: inglés (por defecto), francés, español, neerlandés y alemán. Como ya se ha mencionado, el idioma era configurable a través de los parámetros del applet. Cuando se pulsa sobre

- Configuración: al pulsar sobre esta opción del menú se lanza una página que permite modificar los datos de registro del usuario. Se muestran los datos actuales en campos de textos editables. Al pulsar el botón “Aceptar” se redirige a una página que muestra un mensaje indicando el éxito en la modificación de los datos. Si se deja algún campo sin rellenar, se redirige a otra página que muestra un mensaje de error indicándolo.

- Ayuda: al pulsar sobre esta opción del menú se lanza una página que muestra el manual de usuario, explicando el funcionamiento de la aplicación.

Panel de anotaciones

Implementa la funcionalidad que permite introducir anotaciones: se trata de un panel en la parte izquierda, que contiene un área de texto, donde se muestran las anotaciones de los distintos usuarios sobre una barra con botones que permite navegar entre las anotaciones introducidas por los mismos. Con los botones centrales de la barra se van recorriendo los distintos usuarios y mostrándose sus comentarios para el momento actual de la partida. Con los de los extremos, se avanza al final de la lista de usuarios o al principio, siendo el usuario inicial siempre el usuario actual de la aplicación. Al lado de esta barra de botones aparece un botón que lanza una nueva ventana, un objeto de la clase AnnotationsDialog, que extiende de la clase Dialog e implementa la interfaz ActionListener. En esta ventana de diálogo se implementa un botón de radio por cada uno de los tres tipos de anotación: partida, posición y movimiento. Si se selecciona el botón posición aparecen otros dos botones de radio que permiten seleccionar entre blancas o negras. Los dos combos permiten seleccionar los puntos de inicio y fin de la anotación, pudiendo de esta forma cumplir el requisito de la anotación sobre rangos de fragmentos. Estos combos están desactivados cuando el botón de partida está activado, ya que los rangos sólo aplican para posiciones y movimientos. Adicionalmente, hay un campo de texto donde el usuario puede introducir el texto de la anotación. En la parte inferior hay dos botones, si se pulsa “OK”, se llama al método del servlet que introduce la anotación en la base de datos. Si se pulsa “Cancel”, se cierra la ventana sin añadir la anotación.

6.3 Lectura y conversión de partidas

Los módulos que necesitan leer los archivos PGN son dos: la herramienta de visualización de partidas y la herramienta pobladora de la base de datos.

En el caso de la herramienta de visualización de partidas, el applet original implementaba una clase que crea a partir de una URL un objeto Game, que almacena la información referente a una partida en formato PGN. Este objeto Game es utilizado posteriormente por el applet para dibujar las posiciones en el objeto ChessBoard. Como los archivos PGN se encuentran en un servidor, se utiliza este método ya implementado para acceder a ellos desde el applet a partir de la URL del fichero PGN. El problema es que el nivel de acoplamiento del código que lee el fichero con el código de la herramienta es muy elevado, y está escrito para las necesidades específicas del applet. Por ese motivo resultó muy complicado reutilizar el código para la herramienta pobladora de la base de datos.

Además, también se necesitaba código que permitiera escribir partidas contenidas en objetos Java en ficheros PGN, ya que si un fichero PGN contiene varias partidas se deben separar en ficheros independientes, como se especificó en los requisitos previos. Por consiguiente, para leer los ficheros PGN en la herramienta pobladora de la base de datos y convertirlos en objetos Java se ha utilizado la herramienta Pgnparser, la cual permite leer e interpretar archivos PGN y transformarlos en objetos Java y al contrario, escribir dichos objetos a archivos PGN. Es una herramienta de código libre disponible en el enlace http://sourceforge.net/projects/pgnparser

En la página se señala que el código está distribuido bajo licencia Apache v.2.0, que permite al usuario del software la libertad de usarlo para cualquier propósito, distribuirlo, modificarlo, y distribuir versiones modificadas de ese software. Esa es la razón por la que se ha utilizado dicho código, que de hecho se ha modificado para corregir ciertos comportamientos anómalos y para adecuarlo a las necesidades particulares de nuestra aplicación.

El proceso realizado por esta herramienta consta de dos fases. En primer lugar, la herramienta lee el fichero PGN y devuelve un String con todas las partidas delimitadas por un separador específico, como se muestra en el fragmento de código:

```
while (( line = input.readLine()) != null){
      if(line.startsWith("[") && !lastLine.endsWith("]")){
            contents.append(GAME_SEPARATOR);
      }
      contents.append(line);
      contents.append(System.getProperty("line.separator"));
      lastLine=line;
}
```

Posteriormente se crea una lista de partidas, almacenadas cada una en un objeto Java que contiene toda su información.

```
private List<PGNGame> parseContents(String content) {
            List<PGNGame> games=new ArrayList<PGNGame>();
            String[] gamesString=content.split(GAME_SEPARATOR);
            for(String s:gamesString){
                 String  attributes=s.substring(0,s.lastIndexOf("]")+1);
                  String hits=s.substring(s.lastIndexOf("]")+1,
                      s.length()).trim();
```

```
                if(attributes.length()>0 && hits.length()>0){
                      PGNGame pgn=treatePGNString(attributes, hits);
                      if(pgn!=null games.add(pgn);
                }
          }
          return games;
    }
```

Una vez se ha leído el fichero PGN y se ha obtenido la lista de partidas, para cada una de ellas se crea una entrada en la base de datos a partir de los atributos de la partida. Se crea un fichero PGN por cada una de ellas (en caso de que hubiera varias) en la ruta del servidor correspondiente, ruta que se almacena también en la base de datos.

La escritura del objeto Java al fichero PGN se realiza también mediante un método para tal efecto dentro del parser.

6.4 Herramienta pobladora la base de datos

Como ya se ha mencionado anteriormente, ésta es una herramienta cuyo objetivo es poblar la base de datos con información referente a las partidas. Esta herramienta tiene, por tanto, una funcionalidad limitada y su diseño es básico: una clase principal, que hereda de JPanel, y que lanza una aplicación de escritorio. Y una clase que lanza otra ventana donde cambiar los parámetros de configuración a la base de datos.

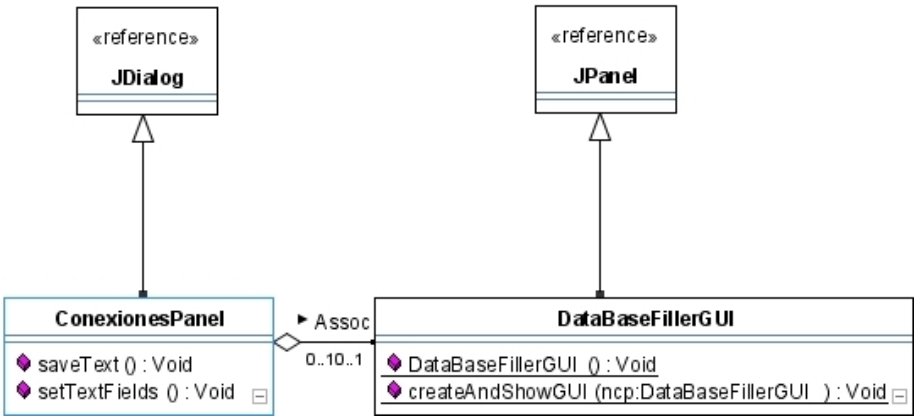


Figura 6.4.1 Diagrama relacional de la herramienta que puebla la base de datos.

El programa implementado está formado por una clase principal DataBaseFillerGUI, que extiende de JPanel y codifica la interfaz gráfica y la consiguiente conexión a la Base de Datos e implementa la interfaz ActionListener. También se utiliza un fichero de configuración, config.xml, en el que se podrán establecer los parámetros de conexión a la base de datos, y la ruta del directorio en el que se almacenarán los ficheros PGN, ruta que debería apuntar al directorio correspondiente dentro de la estructura de ficheros del servidor. La clase que permite la modificación de este fichero es la clase ConexionesPanel, que hereda de la clase JDialog e implementa la interfaz ActionListener.

En la clase DataBaseFillerGUI se crea un objeto un objeto JScrollPane, que contendrá la lista de archivos que se vayan añadiendo, y un panel inferior que contiene los tres JButton que nos permiten realizar todas las operaciones necesarias. El comportamiento de los botones está controlado por la implementación del método actionPerformed de la interfaz ActionListener. También controla el comportamiento de los objetos JCheckBox que se asocian a cada uno de los ficheros añadidos. Por lo tanto, los elementos que lanzan los eventos que recoge este método y sus sendas acciones son los siguientes:

- JButton chooseFile: crea una instancia de la clase JFileChooser con un filtro añadido para limitar la selección de ficheros únicamente a aquellos cuya extensión sea PGN. Los objetos File seleccionados (se habilita multiselección), se añaden a un Vector, y se llama a un método que añade un JCheckBox, con el nombre del fichero como etiqueta, al JPanel contenido en el JScrollPane.

- JButton save: recorre el Vector en el que se han almacenado los ficheros seleccionados. Se obtienen las partidas contenidas dentro del fichero del modo que se explicó en el apartado anterior. Primero se comprueba que no exista una entrada en la base de datos bajo ese nombre de fichero seleccionado. Si existe se lanza un mensaje para que el usuario seleccione otro nombre de fichero. Posteriormente se comprueba que no exista en la base de datos una partida idéntica. Esto se hace comprobando que no exista ninguna partida con el mismo valor de atributos, o etiquetas de partida, en la base de datos. Si existe se lanza un mensaje de error indicando al usuario que dicha partida ya existe en la base de datos. Acto seguido, para cada una de las partidas contenidas en el fichero, se inserta una entrada en la tabla game, con la ruta relativa del fichero, y el contenido de las etiquetas. Si en el fichero hay más de una partida, se crea un nombre del archivo original añadiendo un guión bajo y un número que se incrementa para cada partida consiguiente.

- JCheckBox asociados a los ficheros añadidos: se comprueba si la casilla se ha seleccionado. Si es así, se recorre la lista de ficheros añadidos y se compara por el nombre de la casilla seleccionada, que coincide con los nombre de los ficheros, como ya se ha explicado anteriormente. Cuando se encuentra la coincidencia entre el fichero y la casilla seleccionada, se añade el fichero al Vector que almacena los ficheros con las partidas que se introducirán en la base de datos al lanzarse el evento del botón guardar. Si la casilla en lugar de seleccionarse se deselecciona, se realiza el proceso inverso, se elimina del Vector de ficheros el que coincida con el nombre de la casilla deseleccionada.

- JButton settings: se lanza una instancia de la clase ConexionesPanel, que realiza las operaciones necesarias para modificar el fichero de configuración. Esta clase implementa una serie de campos JTextField y etiquetas JLabel. Estos campos son los que determinan la configuración a la base de datos necesaria para registrar la información de las partidas en la tabla correspondiente. El valor de estos campos es leído del fichero de configuración config.xml. Son los siguientes:

- IP: dirección IP asociada a la conexión a la Base de Datos.
- Puerto: el puerto utilizado por la conexión a la Base de Datos.
- Base Datos: nombre de la propia Base de Datos.
- PGN Folder: directorio en el que se almacenarán los ficheros PGN que contienen las partidas. Este directorio, como se ha explicado anteriormente, debería apuntar al directorio destinado para ese propósito en el servidor. Este parámetro también se puede modificar mediante un objeto JFileChooser

Al pulsarse el JButton "save" de ConexionesPanel se graban los cambios introducidos por el usuario de nuevo en el fichero config.xml y se cierra el JDialog. Si se pulsa el JButton cancel, por el contrario, se cierra el JDialog sin grabar los datos introducidos por el usuario en los campos de texto.

A continuación se describe el contenido del fichero de configuración config.xml:

```
<?xml version="1.0" ?>
<config>
        <url ip="127.0.0.1" port="3306" db="ajedrez"/>
        <user name="root" pwd=""/>
        <pgn path="D:\apache-tomcat-6.0.18\prueba\pgn"/>
 </config>
```

El elemento principal config contiene tres elementos que guardan en sus atributos los parámetros de configuración de la Base de Datos:

- user: elemento que contiene la información relativa al usuario de la Base de Datos en dos atributos:
  - name: nombre de usuario de la Base de Datos.
  - Pwd: contraseña de usuario de la Base de Datos.
- url: elemento que contiene la información relativa a la URL de conexión a la Base de Datos en tres atributos:
  - ip: dirección IP asociada a la conexión a la Base de Datos.
  - port: puerto utilizado por la conexión a la Base de Datos.
  - db: nombre de la propia Base de Datos.
- pgn: elemento que contiene la ruta del directorio en el que se almacenarán los ficheros PGN que contienen las partidas, en el atributo llamado path.

6.5 Gestión de usuarios

En los requisitos iniciales se estipuló la necesidad de un sistema básico de gestión de usuarios. Las funciones principales que deben implementarse son registro, autenticación y modificación de datos de usuario. Para ello se ha implementado una serie de páginas JSP que permiten a un usuario registrarse en la aplicación y acceder a ella. El siguiente esquema muestra la estructura de este módulo.

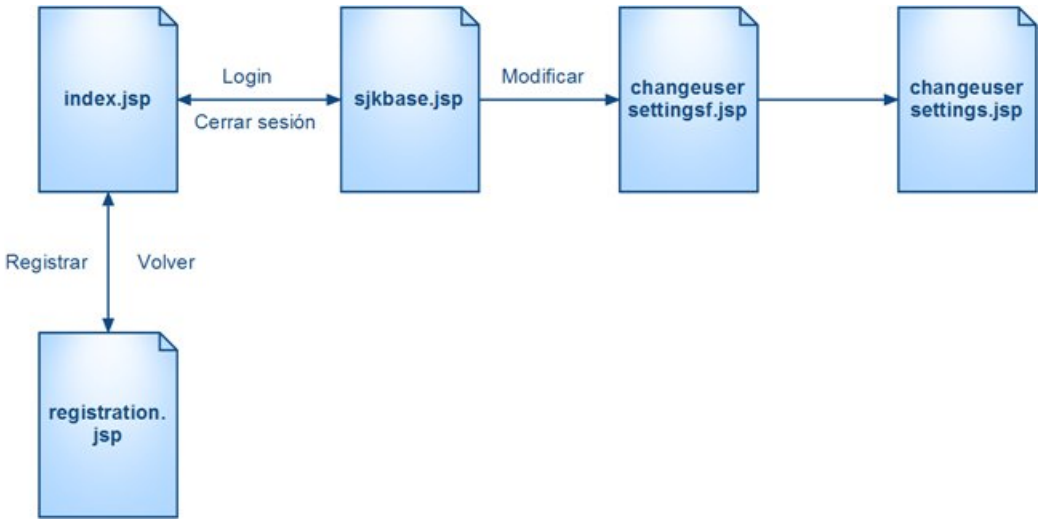


Figura 6.5.1 Estructura de las JSPs en el módulo de gestión.

- index: página de inicio. Permite introducir nombre de usuario y contraseña. Si son erróneas o se deja algún campo sin rellenar, muestra un mensaje de error indicándolo. Si no, redirecciona al usuario a la página sjkbase, que contiene el código que carga el applet. La página index contiene a su vez un hiperenlace a registration.jsp, que permite el registro de usuario.

- registration: muestra una serie de campos que el usuario debe introducir. Si el usuario ya existe o se deja algún campo sin rellenar, muestra un mensaje de error indicándolo. También si la cadena de caracteres de un campo es más larga de lo permitido. Contiene un hiperenlace que permite regresar a index.

- sjkbase: carga el applet, inicializando sus parámetros con los valores determinados por el código de esta página. Los parámetros son los siguientes:

    - Pieces: nombre del archivo de imagen que contiene las piezas de ajedrez.
    - Bgcolor: código de color que se fijará como fondo de la aplicación.
    - Coordinate: indicará si se muestran coordenadas en el tablero o no.
    - Quiz: indicará si se muestra el valor de los movimientos subsiguientes en la partida.

- Language y Country: configuración de lenguaje.

- changeusersettings: esta página se lanza desde una opción del menú del applet y permitirá la modificación de los datos introducidos durante el registro. Para ello se deberá conectar de nuevo con la base de datos para recuperar los datos de registro del usuario en cuestión. Serán presentados de forma que se puedan editar y si el proceso se realiza correctamente, se actualizará la base de datos con las modificaciones introducidas.

- changeusersettingsf: muestra un mensaje indicando que la modificación a sido realizada con éxito, si así ha sido, o bien si se deja algún campo sin rellenar, muestra un mensaje de error indicándolo.

6.6 Servlet intermediario con la base de datos

El servlet sirve como intermediario en la conexión del applet con la base de datos de modo que la seguridad de la comunicación es mayor que si se conectara directamente. El applet necesita conectarse con la base de datos para las siguientes operaciones:

Recuperar una lista de partidas
Cuando el usuario efectúa una búsqueda, introduce una serie de parámetros sobre la que se basará esta búsqueda. El applet crea un objeto List de String con el texto que formará parte de la cláusula "WHERE" de la sentencia SQL que ejecutará la base de datos. Encontramos un ejemplo en el siguiente fragmento de código:

```
List<String> conditions = new ArrayList<String>();
 boolean isAnnotations=false;

if(roundTextField.getText().length()>0)
      conditions.add("g.round='"+roundTextField.getText()+"%'");

if(resultChoice.getSelectedIndex()!=0)
      conditions.add("g.result_id="+resultChoice.getSelectedIndex());

if(userTextField.getText().length()>0){
      conditions.add("a.user_id='"+userTextField.getText()+"'");
      isAnnotations = true;
}
```

El applet entonces conecta con el servlet enviándole un Vector, que contendrá, para esta operación, tres elementos: el primero siempre será el comando por el que el servlet distinguirá de qué operación se trata, "gamesList" en este caso. El segundo será la lista con las condiciones de la sentencia SQL y el tercero es un booleano cuyo valor será "true" siempre que se haya introducido un parámetro de búsqueda relativo a las anotaciones y "false" si no. Esto se debe a que como las anotaciones se encuentran en una tabla distinta a la de los atributos de la partida en la base de datos, la sentencia para recuperarlas debe incluir búsqueda anidada de las dos tablas.

```
Vector param = new Vector();
param.add("gamesList");
param.add(conditions);
param.add(new Boolean(isAnnotations));
```

El servlet recibe el Vector, lee los elementos e invoca al método de la clase que contiene las llamadas a la base de datos, DataBaseConnection. Este método devuelve una lista de objetos GameTags que contienen la información de los atributos de las partidas resultados de la búsqueda, así como la ruta dentro del servidor a cada una de ellas. El servlet crea un objeto ObjectOutputStream, escribe la lista y la envía de vuelta al applet.

```
Vector parameters=(Vector)dataInput;
 String method = (String) parameters.get(0);
if(method.equals("gamesList")){
List<String> conditions = (List<String>) parameters.get(1);
boolean isAnnotations = ((Boolean)parameters.get(2)).booleanValue();
List<GameTags> response =
DataBaseConnection.getInstance().getGamesList(conditions,isAnnotations);

Object obj = response;
res.setContentType("java-internal/" + obj.getClass().getName());
ObjectOutputStream output=new ObjectOutputStream(res.getOutputStream());
output.writeObject(obj);
output.flush();
output.close();
}
```

Recuperar una lista de anotaciones

Cuando el usuario selecciona una partida de la lista devuelta por el servlet, el applet debe recuperar todas las anotaciones relativas a la misma. El applet entonces conecta con el servlet enviándole un Vector, que contendrá, para esta operación, dos elementos: el comando por el que el servlet distinguirá de qué operación se trata, en este caso "annotationList" y el identificador de la partida.

```
idGame = ((GameTags) gamesList.get(index)).getId();
Vector param = new Vector();
param.add("annotationList");
param.add(new Integer(idGame));
annotationList = (List<Annotation>) servletConnection(param);
```

El servlet recibe el Vector, lee los elementos e invoca al método correspondiente de DataBaseConnection. Este método devuelve una lista de objetos Annotation que contienen la información de las anotaciones asociadas a la partida seleccionada. El servlet crea un objeto ObjectOutputStream, escribe la lista y la envía de vuelta al applet.

```
else if(method.equals("annotationList")){
int idGame = ((Integer)parameters.get(1)).intValue();
List<Annotation> response=
DataBaseConnection.getInstance().getAnnotationList(idGame);
Object obj = response;
res.setContentType("java-internal/" + obj.getClass().getName());
```

```
ObjectOutputStream output = new
ObjectOutputStream(res.getOutputStream());
output.writeObject(obj);
output.flush();
output.close();
 }
```

Añadir una anotación

Cuando el usuario añade una anotación, el applet conecta con el servlet enviándole un Vector, que contendrá, para esta operación, dos elementos: el comando por el que el servlet distinguirá de qué operación se trata, en este caso “addAnnotation” y el objeto Annotation que guarda la información relativa a la anotación. El servlet recibe el Vector, lee los elementos e invoca al método correspondiente de DataBaseConnection, pasándole como parámetro el objeto Annotation. Este método ejecuta la inserción en la base de datos. En este caso el servlet no devuelve nada al applet.

```
else if(method.equals("addAnnotation")){
Annotation annotation = (Annotation)parameters.get(1);
DataBaseConnection.getInstance().addAnnotation(annotation);
 }
```

Capítulo 7

Pruebas

La fase de pruebas es una de las más importantes dentro de un proyecto de desarrollo de software. Cuanto más críticas sean la operaciones que realice el programa, más intensivo y más porcentaje de fiabilidad se requiere del proceso de pruebas.

El propósito del presente proyecto, evidentemente no plantea cuestiones críticas, que puedan causar un perjuicio o peligro a los usuarios del mismo. Esto no significa que se deba prescindir de la fase de pruebas, sino que la cobertura de la misma no tiene que ser cercana a 100%, ya que el tiempo dedicado no compensaría el resultado de unas pruebas tan intensivas.

Partiendo de la suposición lógica de que la herramienta de visualización de partidas que se tomó como base ha sido sometida a un conjunto de pruebas que garantizan su correcto funcionamiento, se procederá a la descripción del plan de pruebas al que ha sido sometido el programa con el propósito de verificar su correcto funcionamiento de acuerdo a las especificaciones.

Al diseñar el plan de pruebas se planteó la posibilidad de implementar pruebas de caja blanca, idea que se descartó. El motivo es que el código implementado contiene una cantidad muy reducida de bucles y de condiciones dentro del código. Y como la cobertura deseada no es elevada, se consideró que con pruebas de caja negra ya se alcanzaría un importante nivel de cobertura.

El proceso ha constado de cuatro fases, cada una de las cuales ha sido llevada a cabo en los tres módulos de la aplicación sensibles a ser probados, esto es, la herramienta de visualización de partidas, la herramienta pobladora de la base de datos y la gestión de usuarios. Para esta última se ha excluido la cuarta fase, ya que no resulta procedente. Las fases son las siguientes:

1. Pruebas de caja negra analizando valores límite tanto de entrada como de salida.

2. Identificar clases de equivalencia de datos (entrada y salida) y añadir más pruebas de caja negra para contemplar valores normales.

3. Añadir pruebas basadas en "presunción de error".

4. Prueba de la aplicación en los principales navegadores: Internet Explorer, Mozilla Firefox y Google Chrome.

Cabe destacar que los datos de entrada que el usuario aporta a la herramienta de visualización de partidas tienen un nivel alto de restricción. Esto se debe a que en el

diseño se procuró que para aquéllos parámetros que requiriesen un valor específico, la aplicación implementara un medio de poder seleccionar entre un rango válido de valores, siendo uno de ellos el valor por defecto. Esto es cierto en parámetros de búsqueda tales como el resultado de la partida, o para el tipo de anotaciones (partida, posición, movimiento).

Para el resto de campos en los que se ofrece al usuario libertad para introducir texto, caso extensible al módulo de gestión de usuarios, su contenido no es evaluado por la aplicación, sino que se almacena como tal en la base de datos. Se ha comprobado como clases de equivalencia en estos casos, la longitud de las cadenas de caracteres, ya que en la base de datos hay campos para los que son obligatorias y en todas tienen una longitud máxima.

Respecto a la herramienta pobladora de la base de datos, el dato de entrada es un fichero PGN. Al implementarse un filtro que discrimina los archivos que no son de este tipo, la evaluación se fundamenta en la validez o no del contenido del fichero. Como se analizó anteriormente, el formato PGN es un formato con una estructura bien delimitada, basado en un estándar que especifica claramente los valores válidos de los campos dentro del formato, así como los elementos que los delimitan.

Las pruebas, por tanto, fueron realizadas para todos los módulos como ha sido explicado. Durante el proceso de pruebas se fueron detectando una serie de errores en los distintos módulos que fueron subsanados. Las pruebas fueron repetidas hasta eliminar todos los errores que cubría el conjunto de pruebas diseñado.

Por tanto se puede considerar el resultado del proceso de pruebas como satisfactorio, ya que se ha obtenido una aplicación depurada, que cumple con todos los requisitos funcionales y cuyo funcionamiento se ha comprobado en los principales navegadores del mercado.

Capítulo 8

Conclusiones y Trabajos Futuros

En este apartado se evaluará la consecución de los objetivos planteados inicialmente en el proyecto. Tal y como se planteó en la parte inicial del presente documento, los objetivos fundamentales eran:

- Implementación de una herramienta de visualización de partidas de ajedrez que permitiese añadir anotaciones sobre fragmentos de partidas.
- Implementación de una base de datos donde almacenar los datos de tales partidas.
- Desarrollo de una herramienta para poblar la base de datos.
- Implementación de un mecanismo sencillo de gestión de usuarios.

También se describieron los requisitos tanto funcionales como no funcionales de los distintos módulos de la aplicación.

Con respecto a la base de datos, ante la falta de una base de datos adecuada a los requisitos previos, se optó por el desarrollo de una base de datos propia que contuviera las partidas de ajedrez en formato PGN. La estructura de la base de datos cumple con los requisitos previos, garantizándose la integridad y univocidad de éstos. También se permite el almacenamiento de anotaciones y su asociación con fragmentos específicos dentro de una partida y al usuario que las introduce. Se ha probado que su funcionamiento es correcto, tanto en el almacenamiento como en la recuperación de los datos y que la eficiencia es la deseada.

También se procedió a la implementación de una herramienta para poblar la base de datos con partidas. Se ha probado que su funcionamiento es correcto y que se garantiza la integridad de los datos al introducirlos.

Respecto a la herramienta de visualización de partidas de ajedrez, ya se ha descrito el proceso mediante el cual se seleccionó una herramienta ya existente que cumpliera los requisitos previos en la mayor medida posible. Se seleccionó un applet al que se le realizaron las modificaciones pertinentes para poder cubrir por completo tales requisitos. Así, el applet resultante permite la introducción de anotaciones en la base de datos asociadas a fragmentos de una partida de ajedrez. Las pruebas realizadas constatan el correcto funcionamiento de las operaciones de recuperación de partidas e inserción de anotaciones realizadas por el applet. También se ha trabajado mucho en el aspecto gráfico del applet, que aunque no era uno de los principales objetivos de este proyecto, aporta un mayor valor añadido, convirtiéndolo en una aplicación mucho más amigable y atrayente que el applet original. Para ello se ha tratado de cuidar la manejabilidad y el carácter intuitivo de la herramienta. Además se han añadido funcionalidades extras como la selección de idioma y de piezas de la partida, algo que no estaba entre los requisitos

planteados inicialmente, pero que confiere a la herramienta implementada una singularidad distintiva con respecto a otras posibles herramientas presentes en el mercado. La decisión de implementar estas funcionalidades se tomó ya que gran parte de la estructura que las posibilita estaba implementada en la herramienta original y el tiempo empleado fue muy proporcionado en relación a los beneficios de su inclusión en la herramienta final.

Con respecto a la gestión de usuarios se ha procedido a la implementación de varias operaciones básicas que permiten a un usuario registrarse, acceder a la aplicación y modificar sus datos. En este caso no se han implementado funcionalidades más allá de las estrictamente necesarias, ya que se alejaría del objetivo principal del proyecto, consumiendo una cantidad de tiempo demasiado elevada.

Por tanto la conclusión alcanzada es que los objetivos del proyecto se han cubierto por completo, en cada uno de los módulos en los que se dividió el proyecto inicialmente. Y en el caso del applet, los objetivos incluso se han sobrepasado, habiéndose implementado funcionalidades adicionales que aportan una mayor calidad a la herramienta implementada.

Sin embargo, a pesar de haberse cumplido los objetivos fijados, el trabajo realizado deja las puertas abiertas para futuras ampliaciones e implementación de funcionalidades adicionales.

Una de las posibles ampliaciones en el proyecto podría ser posibilitar que los usuarios pudieran introducir partidas en la base de datos. Ya que la herramienta para poblar la base de datos está implementada, sólo sería necesario implementar una interfaz que permitiera a los usuarios hacer uso de ella, por ejemplo, mediante el propio applet se puede habilitar un botón de selección de fichero, que ese fichero se envíe al servidor y que la herramienta gestionase la inserción en la base de datos.

Como se estableció en los requisitos, la gestión de usuarios es mínima, ya que no era el objetivo de este proyecto. Pero hay mucho margen para ampliar y mejorar la gestión de usuarios. De hecho un proyecto muy interesante que podría surgir del trabajo realizado sería, siguiendo las pautas de la web 2.0, la implementación de una red social con el ajedrez como temática, que utilizara como base la herramienta creada en el presente proyecto. Algunas de sus características y funcionalidades podrían ser las siguientes:

- Incluir más información sobre el usuario, que podría ser solicitada en el momento del registro o completada posteriormente, como por ejemplo lugar y fecha de nacimiento, dirección, teléfono, aficiones, etc.
- Un mecanismo de notificaciones por correo electrónico de eventos relacionados con la aplicación.
- Un mecanismo de restauración de la contraseña en caso de olvido.
- Una funcionalidad que permita jugar con otros usuarios partidas de ajedrez rápidas, en tiempo real, usando el tablero de la aplicación implementada.

- Una funcionalidad que permita jugar con otros usuarios varias partidas de ajedrez simultáneas, más lentas, notificándose al usuario cuando se produzca un movimiento nuevo de uno de los contrincantes.
- Introducir partidas en la base de datos.
- Notificaciones de las acciones realizadas por los contactos del usuario dentro de la red, por ejemplo se podría notificar cada vez que uno de los amigos ha realizado un movimiento en una partida en curso con el usuario, ha añadido una anotación a una partida, ha comenzado una partida nueva, ha añadido una partida a la base de datos, etc.
- Incluir información y noticias relativas al mundo del ajedrez, calendario de eventos, etc.
- Sistema de mensajería básico para interaccionar con el resto de miembros.

La implementación de una red social entera es un trabajo excesivo para un solo proyecto, por lo que la forma de que fuese factible podría ser dividir las funcionalidades en distintos módulos escalables y complementarios que pudieran ser implementados como proyectos separados.

Apéndices

A. Presupuesto

En este apartado del apéndice se procederá a desglosar el proyecto en las distintas fases de su desarrollo y a cuantificar el tiempo empleado para cada una de ellas. También se realizará una estimación del coste real que supondría el desarrollo de este proyecto en un entorno comercial. Debe notarse que en este análisis se excluye el coste de implantación y mantenimiento de las herramientas creadas en un cliente comercial; el análisis sólo cuantifica los gastos asociados al desarrollo y gestión. Se desglosará, por tanto, el coste total del proyecto en tres conceptos distintos: gastos de desarrollo, gastos de gestión y coste del material.

- Gastos de desarrollo: son los gastos totales que generaría la remuneración de una persona encargada del desarrollo de todas las tareas y módulos necesarios para la consecución del proyecto. Las fases en que podría dividirse el proyecto y la estimación del tiempo invertido en cada una de ellas son las siguientes:

    - Recopilación de documentación: recopilación de información relativa a los conceptos concernientes al proyecto, como Web 2.0, tecnologías y herramientas candidatas a ser empleadas en el mismo. Prueba de las distintas aplicaciones de visualización de partidas de ajedrez para analizar sus características. Se estima una cantidad de tiempo invertido de 15 horas.

    - Diseño: diseño de la base de datos. Análisis del diseño de la herramienta de visualización de datos y su posterior. Diseño de la herramienta de inserción de partidas en la base de datos. Diseño del módulo de gestión de usuarios. Se estima una cantidad de tiempo invertido de 15 horas.

    - Configuración: instalación y configuración de las herramientas requeridas para el desarrollo del proyecto: Se estima una cantidad de tiempo invertido de 5 horas.

    - Implementación: implementación de los distintos módulos de la aplicación e interconexión entre los mismo. Implementación de mejoras sucesivas en la herramienta de visualización de partidas. Se estima una cantidad de tiempo invertido de 135 horas.

    - Pruebas: planificación y realización de pruebas para los distintos módulos del proyecto. Se estima una cantidad de tiempo invertido de 15 horas.

    - Documentación: escritura del presente documento detallando el proceso de realización del proyecto. Se estima una cantidad de tiempo invertido de 85 horas.

La suma del tiempo empleado en el desarrollo del presente proyecto está estimada por tanto en 270 horas. Si se asigna a cada hora un precio unitario de 25 euros la hora, que correspondería al precio bruto de la remuneración a un profesional con un perfil de desarrollador, se obtendría un precio total de las labores de desarrollo del proyecto de 6750 euros. Se debe de nuevo mencionar que para simplificar los cálculos el precio de facturación de cada hora de trabajo se estima para la remuneración de un profesional en régimen de autónomo, incluyéndose por tanto en el precio las correspondientes cotizaciones a la Seguridad Social y los gastos asociados al mantenimiento de un puesto físico de trabajo, excluyéndose el material utilizado para el desarrollo del proyecto, ya que se considera un elemento fundamental que requiere un apartado diferenciado en la estimación de gastos.

- Gastos de gestión: son los gastos totales que generaría la remuneración de una persona encargada de la gestión y dirección, tareas realizadas en el presente proyecto por un tutor, que también ha invertido una cantidad de tiempo considerable. Podemos dividir esta labor en las siguientes tareas, con sus respectivas estimaciones de duración:

    - Planteamiento: descripción y planteamiento del proyecto, identificando los objetivos del mismo a partir de un proceso previo de estudio y documentación iniciales. Se estima una cantidad de tiempo invertido de 5 horas.

    - Seguimiento: supervisión de los procesos de diseño, implementación, pruebas y documentación. Se estima una cantidad de tiempo invertido de 40 horas.

    - Gestión: tramitación de documentación administrativa y registros de seguimientos asociados al desarrollo del proyecto: Se estima una cantidad de tiempo invertido de 10 horas.

La suma del tiempo empleado en el la gestión del presente proyecto está estimada por tanto en 55 horas. Si se asigna a cada hora un precio unitario de 40 euros la hora, que correspondería al precio bruto de la remuneración a un profesional con un perfil de gestor, se obtendría un precio total de las labores de gestión de 2200 euros. Se debe de nuevo mencionar que para simplificar los cálculos el precio de facturación de cada hora de trabajo se estima para la remuneración de un profesional en régimen de autónomo, incluyéndose por tanto en el precio las correspondientes cotizaciones a la Seguridad Social y los gastos asociados al mantenimiento de un puesto físico de trabajo.

- Coste de material: para la implementación de este proyecto se ha necesitado un ordenador portátil con conexión a Internet. Se estimará el gasto de alquiler del equipo de 10 euros por día laborable. Se considerará a su vez la dedicación del desarrollador al proyecto en régimen de jornada laboral completa de 40 horas semanales, por lo que las 230 horas corresponderían a 29 días laborables. El coste de alquiler del ordenador portátil supondría por tanto 290 euros. El coste de la conexión

a Internet no se estima como gasto de material, ya que se incluiría dentro de los gastos asociados al mantenimiento del puesto físico de trabajo y por tanto en los gastos de desarrollo.

Por tanto, el presupuesto necesario para el desarrollo de este proyecto sería de 9240 euros, 6750 euros en concepto de gastos de desarrollo, 2200 en gastos de gestión y los restantes 290 euros corresponderían a gastos de material.

B. Manual de instalación y usuario

Aplicación Web 2.0 para la visualización de partidas de ajedrez

Instalación

Para la instalación de la aplicación es necesaria la instalación previa de una base de datos MySQL y de un servidor de aplicaciones. Para este proyecto se ha utilizado un servidor Apache Tomcat 5.5. Y la versión 5.1 de MySQL. La instalación de estas herramientas está documentada en sus respectivas páginas oficiales.

La aplicación se presenta comprimida en el archivo sjkbase.war. Para instalarla, se ha procedido a colocar el archivo war en el directorio correspondiente dentro de la jerarquía del servidor Tomcat, que en este caso es el directorio Tomcat 5.5\webapps. Al arrancarse el servidor, éste lee el archivo war y extrae los archivos contenidos en él dentro de un directorio de mismo nombre, sjkbase, donde se despliega la aplicación.

Hay dos ficheros de configuración del servidor que es necesario modificar son los siguientes:

- META-INF/context.xml: se añade las siguientes líneas, que configuran la conexión a la base de datos:

```
<Context path="/sjkbase" docBase="sjkbase" debug="5" reloadable="true" crossContext="true">
    <Resource name="jdbc/chessDB" auth="Container" type="javax.sql.DataSource" maxActive="100"
    maxIdle="30" maxWait="10000" username="root" password="" driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://127.0.0.1:3306/ajedrez"/>
</Context>
```

- WEB-INF/web.xml: se añade las siguientes líneas, que configuran la conexión a la base de datos y el servlet utilizado en la aplicación:

```
<resource-ref>
    <description>DB Connection</description>
    <res-ref-name>jdbc/chessDB</res-ref-name>
     <res-type>javax.sql.DataSource</res-type>
     <res-auth>Container</res-auth>
 </resource-ref>
 <servlet>
    <servlet-name>AppletConnectionServlet</servlet-name>
    <servlet-class>AppletConnectionServlet</servlet-class>
 </servlet>
 <servlet-mapping>
    <servlet-name>AppletConnectionServlet</servlet-name>
    <url-pattern>/servlet/AppletConnectionServlet</url-pattern>
 </servlet-mapping>
```

Una vez completada la instalación sólo hay que acceder a la aplicación mediante la URL del servidor terminada en /sjkbase.

Manual de usuario

La aplicación se inicia desde la página index.jsp. En esta página aparecen dos campos de texto donde introducir el nombre de usuario y el password. Si son erróneas o se deja algún campo sin rellenar, se muestra un mensaje de error indicándolo. También aparece un enlace bajo el nombre de “Registrarse”. Al pulsar sobre él, el usuario será redirigido a

una nueva página, la cual muestra una serie de campos que el usuario debe introducir. Si el usuario ya existe o se deja algún campo sin rellenar, muestra un mensaje de error indicándolo. Esta página contiene a su vez un enlace que permite regresar a la página inicial.

Una vez introducidos un nombre de usuario y un password correctos, se accede a la herramienta principal, el visor de partidas de ajedrez.

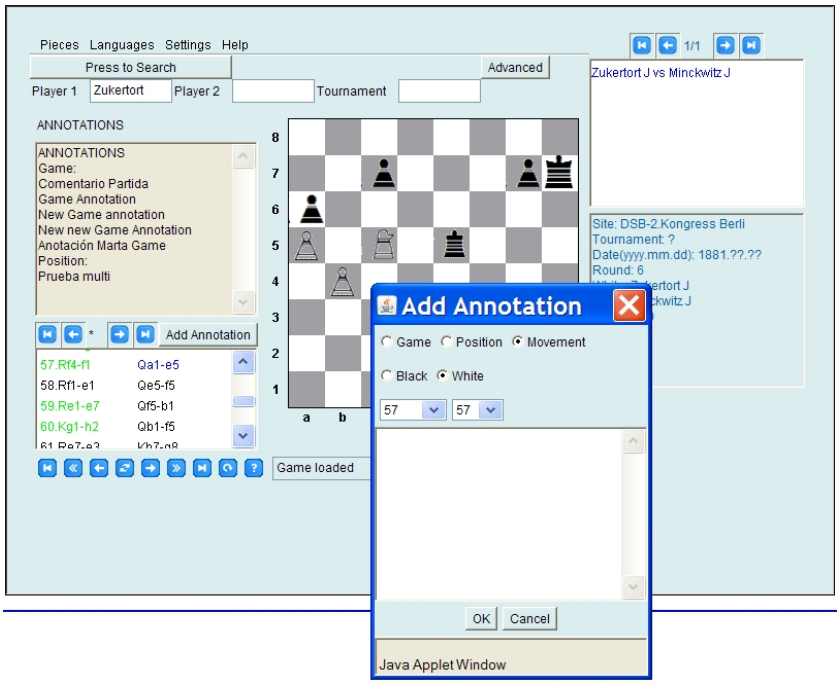


Figura B.1 Ventana de ejemplo de la aplicación.

- **Barra de menú**: situada en la parte superior de la pantalla, la barra de menú permite acceder a una serie de opciones de menú:

  - Piezas: la aplicación ofrece la opción de modificar el tipo de piezas que se dibuja en el tablero de ajedrez. Al pulsar en el menú “Piezas” aparece una lista con los tipos de figuras disponibles. Al seleccionar uno de ellos, se recarga el tablero para mostrarlo con las nuevas piezas seleccionadas.

  - Idiomas: Para cada uno de los idiomas disponibles aparece una entrada en el menú “Idiomas”. Cuando se pulsa, se recarga el contenido de la pantalla y se muestra en el nuevo idioma. Los idiomas disponibles son: inglés, español, francés, alemán, sueco y neerlandés.

  - Configuración: al pulsar sobre esta opción del menú se lanza una página que permite modificar los datos de registro del usuario. Se muestran los datos actuales

je indicando el éxito en la modificación de los datos. Si se deja algún campo sin rellenar, se redirige a otra página que muestra un mensaje de error indicándolo.

- Ayuda: al pulsar sobre esta opción del menú se lanza una página que muestra el manual de usuario, explicando el funcionamiento de la aplicación.

- **Atributos de partida**: al seleccionar una de las partidas resultado de la búsqueda, se muestran en la parte derecha dentro de un cuadro los principales atributos de la partida.

- **Búsqueda**: situado en la parte superior de la pantalla, justo debajo de la barra de menú, este panel muestra una serie de campos sobre los que realizar una búsqueda de partidas. Al acceder a la herramienta, este panel cuenta únicamente con tres campos, los referentes a los jugadores y al torneo, y un botón, de búsqueda avanzada, que muestra el panel con todos los campos de búsqueda permitidos (una serie de campos que permiten realizar una búsqueda a partir de las propiedades obligatorias de una partida en formato PGN, más los relativos a las anotaciones) y en lugar del botón de búsqueda avanzada que había anteriormente, uno de búsqueda simple, el cual al pulsarse devuelve a la versión simplificada del panel. Los campos de búsqueda son, por tanto, los siguientes:

  - Torneo: el nombre del torneo o de la competencia.
  - Lugar: el lugar donde el evento se llevó a cabo
  - Fecha: la fecha de inicio de la partida en formato AAAA.MM.DD.
  - Ronda: La ronda original de la partida.
  - Blancas: El jugador de las piezas blancas.
  - Negras: El jugador de las negras.
  - Resultado: El resultado del juego. Sólo puede tener cuatro posibles valores: "1-0" (las blancas ganaron), "0-1" (Las negras ganaron), "1/2-1/2" (Tablas), o "*" (para otro, ejemplos: el juego está actualmente en disputa o un jugador falleció durante la partida).
  - Usuario: usuario que ha introducido la etiqueta.
  - Tipo: El tipo de anotación. Sólo puede tener valores: partida, posición y movimiento.
  - Texto: contenido de la anotación realizada por el usuario.

  Para los campos torneo, lugar, fecha, ronda, blancas y negras, se proporciona un campo de texto que permite al usuario introducir libremente el texto de búsqueda. En la etiqueta que acompaña al campo fecha se indica el formato que debe tener la fecha. Para los campos blancas y negras se ha fijado el texto como “jugador 1” y “jugador 2”, con la intención de que se recuperen las partidas para las que el nombre de sus jugadores coincida con cualquiera de estos dos parámetros de búsqueda, independientemente de cuál de los dos jugara con blancas y cuál con negras. Para el campo resultado, por el contrario, se habilita una caja de selección o combo, que

permite elegir entre los únicos cuatro valores permitidos. Para los campos usuario y texto, se proporciona un campo de texto que permite al usuario introducir libremente el texto de búsqueda. Para el campo tipo, por el contrario, se habilita una caja de selección, que permite elegir entre los únicos tres valores permitidos.

Al rellenar alguno de los campos y pulsar el botón “Buscar”, la aplicación realiza una búsqueda en la base de datos de las partidas que se ajusten a los parámetros de búsqueda introducidos y muestra una lista de los resultados obtenidos en el cuadro adyacente, en la parte derecha. Este cuadro muestra la lista de partidas en grupos de cinco. Si hubiera más de cinco partidas que coincidan con los parámetros, la barra de botones encima del cuadro permite navegar por la lista completa de partidas recuperadas. Si se selecciona una de las partidas, haciendo clic, se cargará la partida, mostrándose en el cuadro inmediatamente inferior la información contenida en las siete etiquetas básicas de la partida y en el cuadro inferior de la parte izquierda de la pantalla, la lista de movimientos de la partida.

- **Tablero de ajedrez**: ocupa la parte central de la pantalla y despliega la posición de las piezas para cada movimiento de la partida. La forma de cambiar la posición de las piezas del tablero es, bien pulsando uno de los botones de la barra de movimientos, bien pinchando sobre cualquiera de los movimientos desplegados en el cuadro que los muestra todos. El tipo de piezas puede cambiarse, como se ha mencionado anteriormente, mediante el menú “Piezas” de la barra de menú en la parte superior de la pantalla.

- **Anotaciones**: se muestran en un panel situado en la parte izquierda, que contiene un área de texto, donde se muestran el contenido de las anotaciones, una barra con botones en su parte inferior, para navegar entre las anotaciones introducidas por distintos usuarios, incluido el usuario actual y un botón que lanza una nueva ventana. Esta nueva ventana incorpora un área de texto editable, donde el usuario puede introducir el texto de la anotación, además de una serie de botones de radio y dos combos. Hay un botón de radio por cada uno de los tres tipos de anotación: partida, posición y movimiento. Si se selecciona el botón posición aparecen otros dos botones de radio que permiten seleccionar entre blancas o negras. Los dos combos permiten seleccionar los puntos de inicio y fin de la anotación. Estos combos están desactivamos cuando el botón de partida está activado, ya que los rangos sólo aplican para posiciones y movimientos. Al pulsar el botón “OK” en esta nueva ventana se introduciría la anotación, mientras que al pulsar el botón “Cancel” se cerraría la ventana sin introducir anotación alguna.

- **Movimientos**: muestra la lista de todos los movimientos de la partida seleccionada. Está situado en la parte inferior de la pantalla, justo debajo de las anotaciones. Cuando un movimiento o una posición contienen un comentario por parte de alguno de los usuarios, ese movimiento o posición aparecen en color verde, facilitándose así al usuario el acceso a los fragmentos de la partida con anotaciones.

Barra de estado: campo en la parte inferior de la pantalla donde se van mostrando los mensajes asociados a las acciones realizadas por el usuario.

Herramienta para poblar la base de datos

Instalación

La herramienta aparece comprimida en un fichero tipo zip. La única instalación que se requiere es la descompresión de este fichero. Este fichero contiene una carpeta llamada lib, con las librerías necesarias para ejecutar la aplicación, en fichero config.xml, que almacena los datos de configuración de la base de datos, el archivo datafiller.jar, que contiene el código compilado de la aplicación, y el fichero run.exe. Al ejecutar este último se lanza la aplicación. La versión de java que se debe utilizar es jre1.6.0.

Manual de usuario

Al iniciarse la aplicación, se nos presenta una ventana con el texto "Lista de ficheros PGN seleccionados:" y un espacio vacío. En la parte inferior se presenta una barra con tres botones.

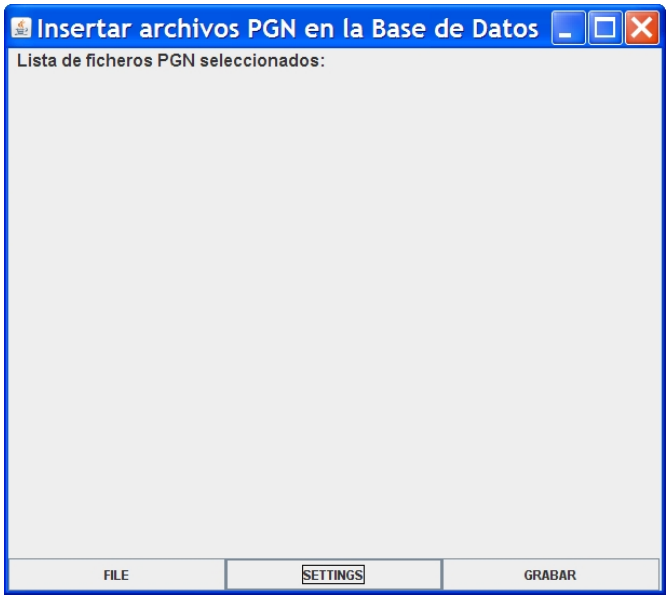


Figura B.2 Ventana inicial de la aplicación DataBaseFiller.

Las únicas acciones posibles desde esta ventana principal al inicio es pulsar uno de estos botones.

- Botón "File": al pulsar este botón aparecerá una ventana de selección de archivos. Esta ventana nos muestra el sistema de archivos de nuestro equipo y nos permite navegar por las diferentes carpetas existentes, aunque sólo nos mostrará aquellos archivos cuya extensión sea PGN o pgn, y por tanto podrán ser seleccionados, bien

pulsando sobre el archivo y posteriormente en el botón abrir, bien con un doble clic sobre el archivo.

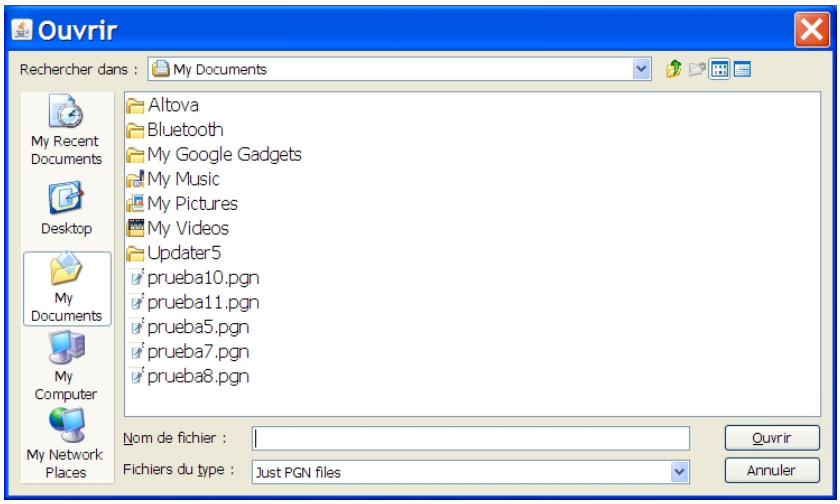


Figura B.3 Ventana de selección de archivos.

También se permite la opción de seleccionar varios archivos al mismo tiempo

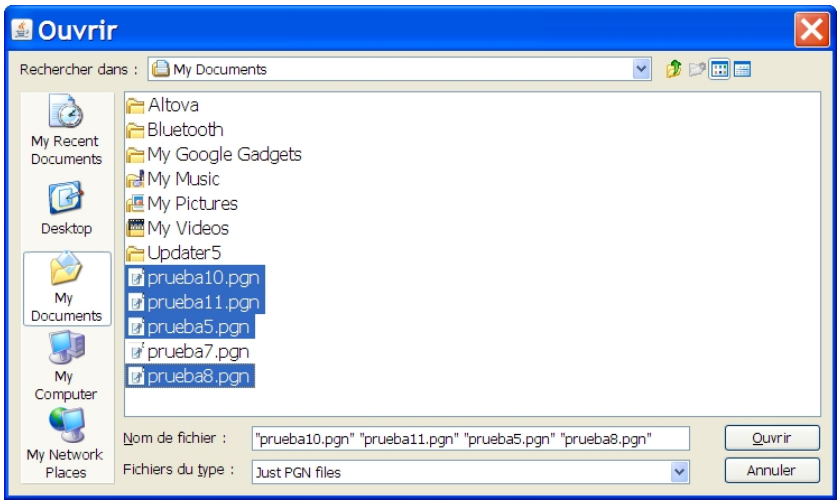


Figura B.4 Ventana de selección de archivos, con varios de ellos marcados.

Una vez seleccionados uno a varios de los ficheros PGN, estos se mostrarán en la ventana principal, con una casilla seleccionable a la izquierda del nombre del archivo.

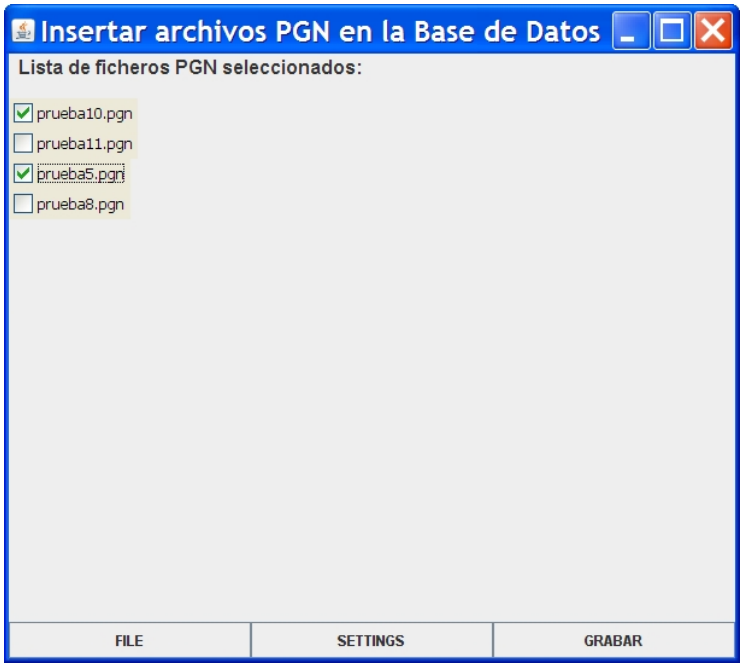


Figura B.5 Ventana principal de la aplicación, con una lista de los archivos añadidos y dos de ellos seleccionados.

- Botón “Grabar”: este botón permite grabar la información relativa a las partidas en la Base de Datos. Una vez pulsados, si no hay ningún error, los archivos desaparecen de la ventana principal.

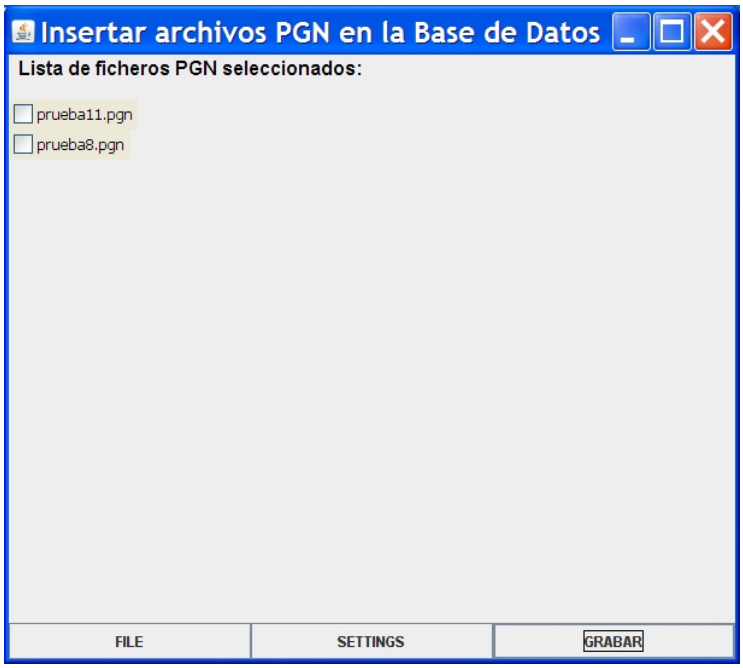


Figura B.6 Ventana principal de la aplicación, con una lista de los archivos añadidos habiendo desaparecido los dos archivos grabados.

Si en la Base de Datos ya existiese una partida exacta a la que se intenta almacenar, lo que es lo mismo, una partida con el mismo valor en las siete etiquetas obligatorias del formato PGN, un mensaje de error aparecerá en la pantalla por cada una de las partidas que sea exacta indicándolo, por lo que la parida no podrá ser grabada y desaparecerá de la pantalla principal.

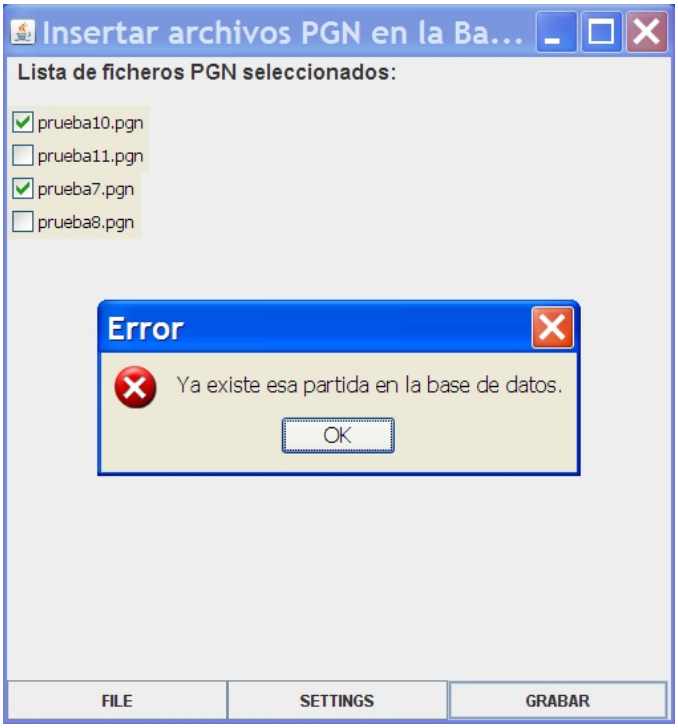


Figura B.7 Mensaje de error al intentar grabar una partida ya existente en la Base de Datos.

Si se intenta grabar una partida que, aunque distinta a todas las contenidas en la Base de Datos, coincide en nombre con alguna de ellas, aparecerá un mensaje una ventana que permitirá introducir un nuevo nombre para el fichero.

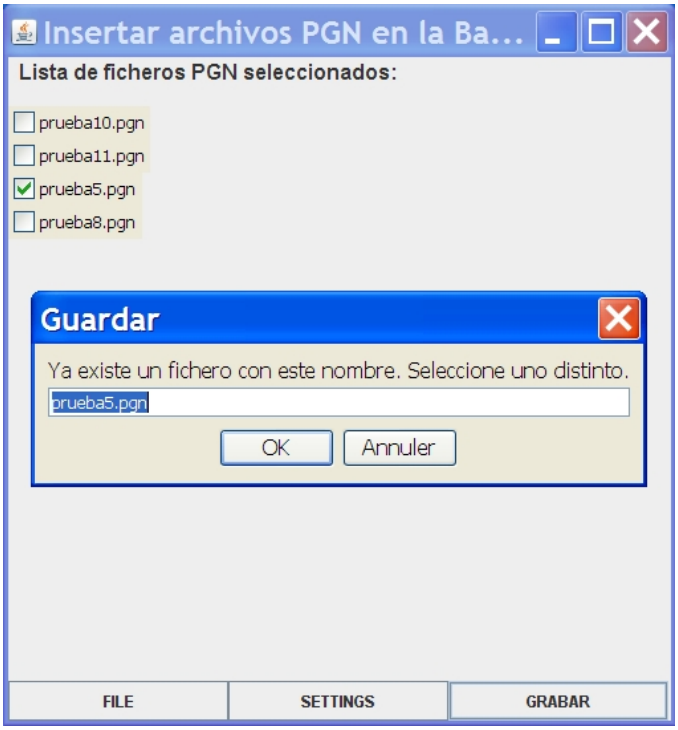


Figura B.8 Ventana auxiliar que permite especificar un nombre distinto para los archivos, cuando ya existe algún archivo con el mismo nombre.

- Botón "Settings": al pulsar este botón aparece una nueva pantalla con varios campos de texto que nos permiten modificar los parámetros de configuración a la base de datos. El usuario puede pinchar sobre cada uno de los campos y modificar el contenido si lo hubiera.

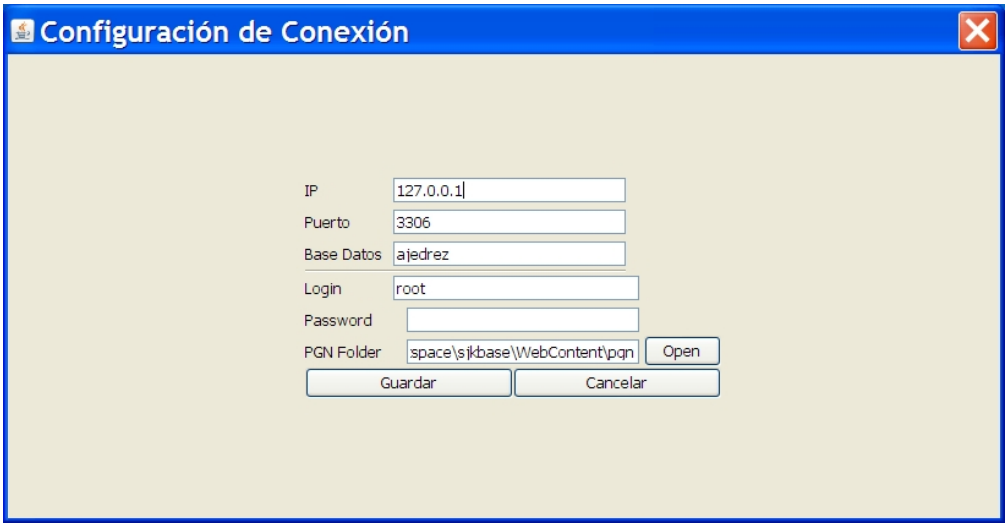


Figura B.9 Ventana que permite modificar el valor de los parámetros de configuración de la conexión a la Base de Datos .

Nótese que el único campo que tiene un botón asociado es la que especifica la ruta donde se guardan los archivos PGN en el servidor. Al pulsar dicho botón aparece una ventana de selección de archivos que nos permite escoger el directorio de entre el sistema de archivos de nuestro equipo.

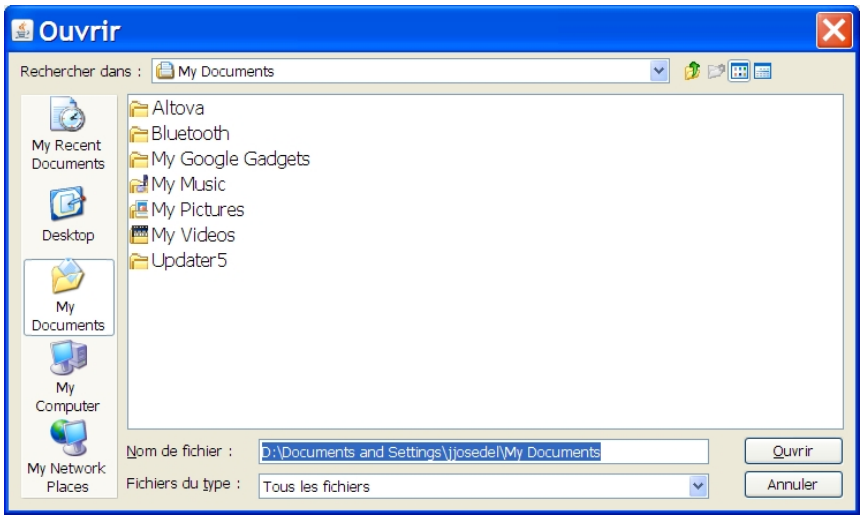


Figura B.10 Ventana de selección de archivos, que nos permite seleccionar el directorio en el que se guardarán los archivos PGN.

En la parte inferior de la ventana vemos dos botones, aceptar y cancelar. Al pulsar sobre el botón aceptar, el contenido introducido por el usuario en los campos se guarda en el fichero config.xml y se cierra la ventana. Si, por el contrario, se pulsa sobre el botón cancelar, la ventana se cierra sin guardar el contenido introducido por el usuario en los campos de texto.

Referencias

[1] Rainie, L. “Tagging.” Pew Internet & American Life Project, 31 January 2007. http://www.pewtrusts.org/our_work_report_detail.aspx?id=21170
[2] "Web 2.0". Artículo de Wikipedia en español. Consultado el 11 de julio de 2009. http://es.wikipedia.org/wiki/Web_2.0
[3] Tim O'Reilly «What Is Web 2.0. Design Patterns and Business Models for the Next Generation of Software», http://oreilly.com/web2/archive/what-is-web-20.html
[4] La Web 2.0. El valor de los metadatos y de la inteligencia colectiva, de Xavier Ribes en "Telos. Cuadernos de Comunicación e Innovación", n. 73 (2007) de la Fundación Telefónica.
[5] "Notación portable de juego". Artículo de Wikipedia en español. Consultado el 13 de julio de 2009. http://es.wikipedia.org/wiki/Notaci%C3%B3n_portable_de_juego
[6] O'd Alexander, C. H. and Beach, T. J., Learn Chess: A Complete Course, 3ª ed., MacMillan Publishing Company, 2003. ISBN 1-857-44115-X.
[7] "JavaScript". Artículo de Wikipedia en español. Consultado el 12 de julio de 2009. http://es.wikipedia.org/wiki/JavaScript
[8] "Applet Java". Artículo de Wikipedia en español. Consultado el 12 de julio de 2009. http://es.wikipedia.org/wiki/Applet_Java
[9] "MySql". Artículo de Wikipedia en español. Consultado el 12 de julio de 2009. http://es.wikipedia.org/wiki/MySql
[10] Kofler, Michael, The Definitive Guide to MySQL 5, 3.ª ed., Apress, 2005. ISBN 1-59059-535-1.
[11] "Postgresq ". Artículo de Wikipedia en español. Consultado el 12 de julio de 2009. http://es.wikipedia.org/wiki/Postgresql
[12] "Oracle". Artículo de Wikipedia en español. Consultado el 12 de julio de 2009. http://es.wikipedia.org/wiki/Oracle
[13] "Microsoft SQL Server". Artículo de Wikipedia en español. Consultado el 12 de julio de 2009. http://es.wikipedia.org/wiki/Microsoft_SQL_Server
[14] "Adaptive Server Enterprise". Artículo de Wikipedia en español. Consultado el 12 de julio de 2009. http://es.wikipedia.org/wiki/Adaptive_Server_Enterprise
[15] "Interbase". Artículo de Wikipedia en español. Consultado el 13 de julio de 2009. http://es.wikipedia.org/wiki/Interbase
[16] "Firebird ". Artículo de Wikipedia en español. Consultado el 13 de julio de 2009. http://es.wikipedia.org/wiki/Firebird
[17] "Swing (biblioteca gráfica)". Artículo de Wikipedia en español. Consultado el 13 de julio de 2009. http://es.wikipedia.org/wiki/Swing_(biblioteca_gr%C3%A1fica)
[18] Eckstain, Robert, Loy, Marc y Word, Dave, JAVA Swing, 2.ª ed., O'Reilly Media, Inc., 2002. ISBN 0-59600-408-7.
[19] "Abstract Window Toolkit". Artículo de Wikipedia en español. Consultado el 13 de julio de 2009. http://es.wikipedia.org/wiki/Abstract_Window_Toolkit
[20] "Licencia de software". Artículo de Wikipedia en español. Consultado el 13 de julio de 2009. http://es.wikipedia.org/wiki/Licencia_de_software
[21] Schildt, Herbert, Java – A Beginner’s Guide, 3ª ed., McGraw-Hill, 2005. ISBN 0-07-146650-9.

[22] "Jakarta_Tomcat". Artículo de Wikipedia en español. Consultado el 16 de septiembre de 2009. http://es.wikipedia.org/wiki/Licencia_de_software http://es.wikipedia.org/wiki/Jakarta_Tomcat
[23] Falkner, Jayson and Jones, Kevin, Servlets and JavaServer Pages: The J2EE Technology Web Tier, 1ª ed., Addison-Wesley Professional, 2003. ISBN 0-321-13649-7.

Glosario de acrónimos

- ACID: Atomicity, Consistency, Isolation, Durability
- AJAX: Asynchronous JavaScript and XML
- ANSI: American National Standards Institute
- API: Application Programming Interface
- ASE: Adaptive Server Enterprise
- AWT: Abstract Window Toolkit
- BLOB: Binary Large Objects
- BSD: Berkeley Software Distribution
- CIDR: Classless Inter-Domain Routing
- CDDL: Common Development and Distribution License
- CSS: Cascading Style Sheets
- DDL: Data Definition Language
- DML: Data Manipulation Language
- eCos: Embedded Configurable Operating System
- EPD: Extended Position Description
- EUPL: European Union Public Licence
- FEN: Forsyth-Edwards Notation
- FOAF: Friend Of A Friend
- GIS: Geographic Information System
- GNU GPL: GNU General Public License
- GUI: Graphic User Interface
- HP-UX: Hewlett Packard UniX
- HTML: HyperText Markup Language
- IP: Internet Protocol
- J2SDK: Java 2 Software Development Kit
- JAR: Java Archive
- JDBC: Java Database Connectivity
- JFC: Java Foundation Classes
- JVM : Java Virtual Machine
- JCC: JavaScript Client Communication
- JRE: Java Runtime Environment
- JSON: JavaScript Object Notation
- JSP: Java Server Pages
- LDAP: Lightweight Directory Access Protocol
- MAC: Media Access Control Address
- Mac OS: Macintosh Operating System
- MIT: Massachusetts Institute of Technology
- NEA: Notación Estándar Algebraica
- ODBC: Open Database Connectivity
- OLE DB: Object Linking and Embedding Database
- OWL: Web Ontology Language

- PGDG: PostgreSQL Global Development Group
- PGN: Portable Game Notation
- PGZ: Portable Game Notation Zipped
- POWDER: Protocol for Web Description Resources
- RDBMS: Relational database management system
- REST: Representational state transfer
- RSS: Really Simple Syndication
- SOAP: Simple Object Access Protocol
- SPARQL: SPARQL Protocol and RDF Query Language
- SQL: Structured Query Language
- SSL: Secure Sockets Layer
- STR: Seven Tag Roster
- TCP/IP: Transmission Control Protocol / Internet Protocol
- UDF: User-Defined Function
- URL: Uniform Resource Locator
- XFN: XHTML Friends Network
- XHTML: Extensible Hypertext Markup Language,
- XML: Extensible Markup Language
- XUL: XML User Interface Language
- W3C: World Wide Web Consortium
- WAR: Web Application Archive